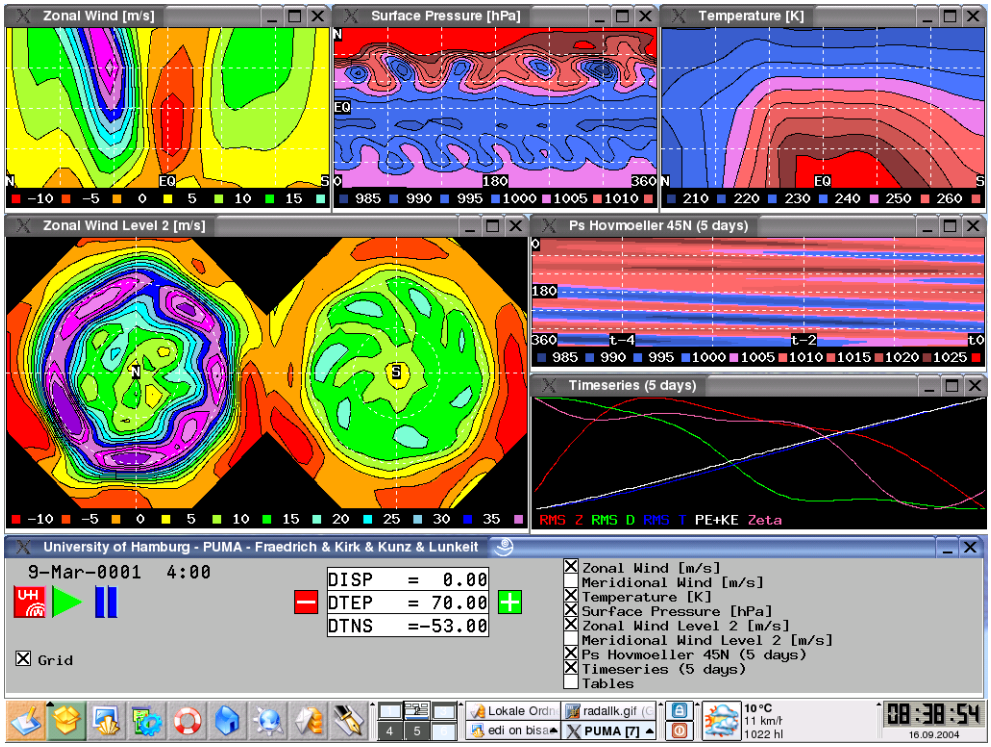
University of Hamburg

PUMA



User's Guide

Version 15.0

Klaus Fraedrich
Simon Blessing
Edilbert Kirk
Torben Kunz
Ute Luksch
Frank Lunkeit
Frank Sielmann

The PUMA User's Guide is a publication of the
Theoretical Meteorology at the Meteorological Institute of
the University of Hamburg.

Address:

Prof. Dr. Klaus Fraedrich
Meteorological Institute
University of Hamburg
Bundesstrasse 55
D-20146 Hamburg

Contact:

Klaus.Fraedrich@zmaw.de
Frank.Lunkeit@zmaw.de
E.Kirk@gmx.de

Contents

1 Installation **5**
- 1.1 Quick Installation . . . 5
- 1.2 Most15 directory . . . 5
- 1.3 Model build phase . . . 6
- 1.4 Model run phase . . . 7
- 1.5 Running long simulations . . . 7

2 Introduction **9**
- 2.1 Training of junior scientists and students . . . 9
- 2.2 Compatibility with other models . . . 9
- 2.3 Scientific applications . . . 10
- 2.4 Requirements . . . 10
- 2.5 History . . . 11

3 Modules **13**
- 3.1 fftmod.f90 / fft991mod.f90 . . . 13
- 3.2 fileseek.f90 / fseek64.f90 / fseek.f90 . . . 14
- 3.3 guimod.f90 / guimod_stub.f90 . . . 15
- 3.4 legmod.f90 . . . 16
- 3.5 mpimod.f90 / mpimod_stub.f90 . . . 17
- 3.6 puma.f90 . . . 19
- 3.7 pumamod.f90 . . . 21
- 3.8 resmod.f90 . . . 22
- 3.9 restartmod.f90 . . . 23

4 Parallel Program Execution **25**
- 4.1 Concept . . . 25
- 4.2 Parallelization in Gridpoint Domain . . . 25
- 4.3 Parallelization in Spectral Domain . . . 26
- 4.4 Synchronization points . . . 26
- 4.5 Source code . . . 27

5 Graphical User Interface **29**
- 5.1 Graphical user interface (GUI) . . . 29
- 5.2 GUI configuration . . . 31
  - 5.2.1 Array . . . 32
  - 5.2.2 Plot . . . 32
  - 5.2.3 Palette . . . 33
  - 5.2.4 Title . . . 33
  - 5.2.5 Geometry . . . 33

6 Postprocessor Pumaburner **35**
6.1 Introduction . . . 35
6.2 Usage . . . 35
6.3 Namelist . . . 36
6.4 HTYPE . . . 36
6.5 VTYPE . . . 36
6.6 MODLEV . . . 36
6.7 hPa . . . 37
6.8 LATS and LONS . . . 37
6.9 MEAN . . . 37
6.10 Format of output data . . . 38
6.11 SERVICE format . . . 38
6.12 HHMM . . . 38
6.13 HEAD7 . . . 38
6.14 MARS . . . 39
6.15 MULTI . . . 39
6.16 Namelist example . . . 39
6.17 Troubleshooting . . . 39

7 Graphics **41**
7.1 Grads . . . 41
7.2 Vis5D . . . 44

8 Model Dynamics **47**
8.1 Model equations and numerics . . . 47
8.2 Parameterizations . . . 49
8.2.1 Friction . . . 49
8.2.2 Diabatic heating . . . 49
8.2.3 Diffusion . . . 50
8.3 Scaling of Variables . . . 52
8.4 Vertical Discretization . . . 52
8.5 PUMA Flow Diagram . . . 53
8.6 Initialization . . . 53
8.7 Computations in spectral domain . . . 54

A List of Constants and Symbols **59**

B PUMA Codes for Variables **63**

C Namelist **65**

Chapter 1

Installation

The whole package, containing the models "Planet Simulator" and ""PUMA" along with "MoSt", the model starter, comes in a single file named "Most(n).tgz" with (n) specifying a version number. The following subsection shows the commands to use assuming the version number is 15.

1.1 Quick Installation

```
tar -zxvf Most15.tgz
cd Most15
./configure.sh
./most.x
```

If your tar-command doesn't support the "-z" option (e.g. on Sun UNIX) type instead:

```
gunzip Most15.tgz
tar -xvf Most15.tar
cd Most15
./configure.sh
./most.x
```

If this sequence of commands produces error messages, consult the "FAQ" (Frequent Asked Questions) and README files in the Most15 directory. They are in plain text, that can be read with the command "more" or opened by a text editor.

1.2 Most15 directory

```
home/Most15> ls -lG

-rw-r--r-- 1   1548 cc_check.c          <- Used by configure.sh
-rwxr-xr-x 1     57 cleanplasim         <- Delete run, bld and bin for PLASIM
-rwxr-xr-x 1     51 cleanpuma           <- Delete run, bld and bin for PUMA
drwxr-xr-x 2   4096 common              <- Topography files
-rwxr-xr-x 1   3911 configure.sh        <- The configure script
-rw-r--r-- 1    308 csub.c              <- Currently unused
-rw-r--r-- 1    234 f90check.f90        <- Used by configure.sh
-rw-r--r-- 1   3033 FAQ                 <- Frequently Asked Questions
drwxr-xr-x 2   4096 images              <- Directory for images
```

```
-rw-r--r-- 1    154 makecheck           <- Used by configure.sh
-rw-r--r-- 1     85 makefile            <- Used to "make" most.x
-rw-r--r-- 1 107844 most.c              <- Source for MoSt (Model Starter)
-rw-r--r-- 1   6399 NEW_IN_VERSION_15   <- New in this version
drwxr-xr-x 8   4096 plasim              <- Planet Simulator directory
drwxr-xr-x 2   4096 postprocessor       <- Postprocessor directory
drwxr-xr-x 8   4096 puma                <- PUMA directory
-rw-r--r-- 1    839 README              <- Read this first
-rw-r--r-- 1    191 README_MAC_USER     <- Notes for MAC user
-rw-r--r-- 1    698 README_WINDOWS_USER <- Notes for Windows user
```

The directory structure must not be changed. Even empty directories must be kept as they are, because the Most program relies on their existence!

For each model, currently "Planet Simulator" and "PUMA", there exists a directory (plasim or puma) with the following subdirectories:

```
Most15/puma> ls -lg

drwxr-xr-x  2   128  bin   <- model executables
drwxr-xr-x  2  1824  bld   <- build directory
drwxr-xr-x  2   280  dat   <- initial and boundary data
drwxr-xr-x  2    80  doc   <- documentation, user's guide, reference manual
drwxr-xr-x  2   928  run   <- run directory
drwxr-xr-x  2  1744  src   <- source code
```

After installation only "dat", "doc" and "src" contain files. All other directories are empty.

Running "Most" to setup a model configuration and define an experiment uses the directories in the following manner:

1.3 Model build phase

Most writes an executable shell script to the "bld" directory and ithen executes it. First, it copies all necessary source files from "src" to "bld" and modifies them according to the selected parameter configuration. Modification of source code is necessary for vertical and horizontal resolution change, and when using more than 1 processor (parallel program execution). The original files in the "src" directory are not changed by Most.

The program modules are then compiled and linked using the "make" command, also issued by Most. Most provides two different makefiles: one for the single CPU version and the other or the parallel version (using MPI, Message Passing Interface). The resolution and CPU parameters are coded into the filename of the executable, in order that there are different names for different versions. E.g. the executable "most_puma_t21_l10_p2.x" is an executable compiled for a horizontal resolution of T21, a vertical resolution of 10 levels and 2 CPU's.

The executable is copied to the model's "bin" directory at the end of the build. Each time Most is used to setup a new experiment it checks the "bin" directory for a matching executable. If it exists, it is used without rebuilding, otherwise a new executable with the selected parameters is created. Rebuilding may be forced by using the cleanpumacommand in the Most directory. The build directory is not cleared after usage. The user may want to modify the makefile or the build script for his own purposes and start the building directly by executing "most_puma_build". For permanent user modifications the contents of the "bld" directories have to be copied elsewhere, because each usage of Most overwrites the contents of "bld".

1.4 Model run phase

After building the model with the selected configuration, Most writes or copies all necessary files to the model's "run" directory. These are the executable, initial and boundary data, namelist files containing the parameter and finally the run script itself. Depending on the exit from Most, either "Save & Exit" or "Run & Exit", the run script is started from Most and takes control of the model run. A checkmark on GUI invokes also the the Graphical User Interface for user control and display of variables during the run. Again all contents of the "run" directory are subject of change for the user. But it would be wise to keep changed run setups in other, user created directories, because each usage of Most will overwrite the contents of the run directory. Another concept could be to rename user changed files, because Most always generated files starting with "most_" and leaves other files untouched.

1.5 Running long simulations

For long simulations make a new directory on a filesystem, that has enough free disk space to store the results. You may use the "df" command to check filesystems.

Hint 1: Don't use your home directory if there are filequotas. Your run may crash due to file quota exceeded.

Hint 2: Use a local disk, not NFS mounted filesystems if possible. The model runs much faster writing output to local disks.

Example:

- cd Most15
- ./most.x
- Select model and resolution
- Switch GUI off
- Switch Output on
- Edit number of years to run
- Click on "Save & Exit"
- Make a directory, e.g. mkdir /data/longsim
- cp puma/run/* /data/longsim
- cd /data/longsim
- edit most_puma_run for experiment name
- edit namelist files if necessary
- start simulation with most_puma_run &

Chapter 2

Introduction

The **P**ortable **U**niversity **M**odel of the **A**tmosphere **(PUMA)** is based on the multi-level spectral model **SGCM** (**S**imple **G**lobal **C**irculation **M**odel) described by Hoskins and Simmons (1975) and James and Gray (1986). Originally developed as a numerical prediction model, it was changed to perform as a circulation model. For example, James and Gray (1986) studied the influence of surface friction on the circulation of a baroclinic atmosphere, James and James (1992) and James et al.(1994) investigated ultra-low-frequency variability, and Mole and James (1990) analyzed the baroclinic adjustment in the context of a zonally varying flow. Frisius et al.(1996) simulated an idealized storm track by embedding a dipole structure in a zonally symmetric forcing field and Lunkeit et al. (1998) investigated the sensitivity of GCM scenarios by an adaption technique applicapable to SGCMs. The Storm track dynamics and low frequency variability was investigated by [Fraedrich et al., 2005].

PUMA is introduced with following aims in mind: Training of junior scientists, compatibility with ECHAM (**E**uropean **C**entre - **HAM**burg) and scientific applications.

2.1 Training of junior scientists and students

PUMA contains only the main processes, necessary to simulate the atmosphere. The source code is short and clearly arranged. A student can learn to work with PUMA within few weeks, whereas full size GCMs require a team of specialists for maintenance, experiment design and diagnostics.

2.2 Compatibility with other models

PUMA is designed to be compatible with other circulation models like Planet Simulator and ECHAM. The same triangular truncation is employed, and analogous transformation techniques like the Legendre- and Fast-Fourier transformation are used. The postprocessor **Pumaburner** differs from ECHAM's **Afterburner** only in respect to the format of the model's raw data which overcomes some problems of ECHAM's data format. PUMA uses a more compact though more precise format compared to the GRIB (**GRI**dded **B**inary), that is used for ECHAM output. The Pumaburner supports the output formats SERVICE or NetCDF. All diagnostics and graphics software, that is applied on ECHAM/Afterburner data can be used for PUMA/Pumaburner in exactly the same way.

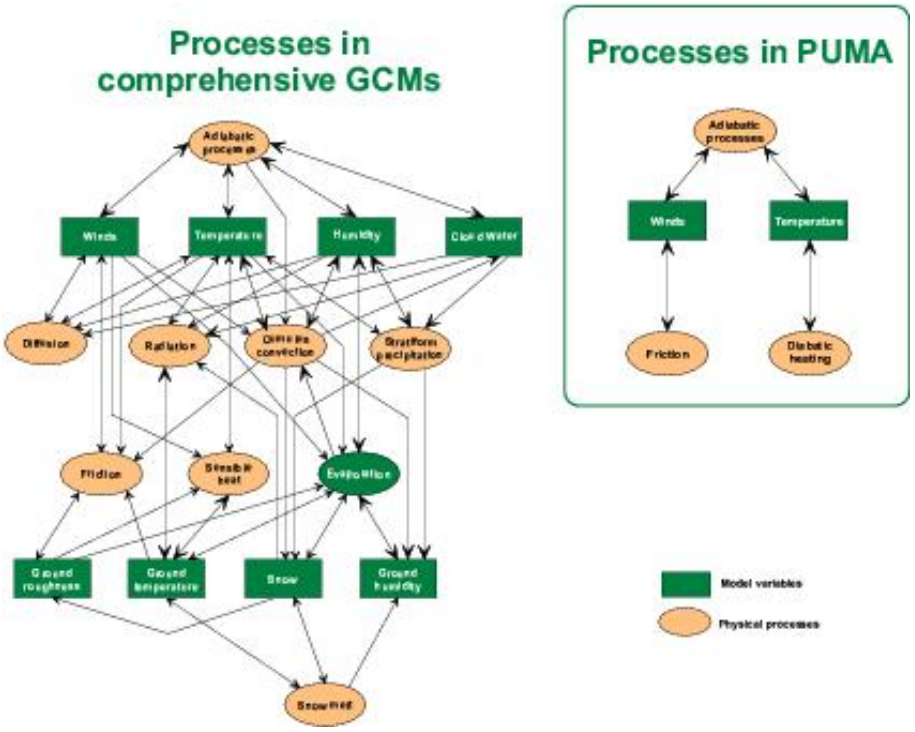


Figure 2.1: Processes in ECHAM (left) and PUMA (right)

2.3 Scientific applications

The PUMA code is the dynamical core of a GCM forced by Newtonian cooling and Rayleigh friction, such as proposed by Held and Suarez (1994) to evaluate the dynamical cores of GCMs. It forms the basis for various applications:

- The code can be utilized to build and test new numerical algorithms (like semi-Langrangian techniques).
- Idealized experiments can be performed to analyze nonlinear processes arising from internal atmospheric systems (life cycles, etc.).
- Data assimilation techniques can be incorporated to interpret results from GCM simulations or observations.

Figure 2.1 (left) demonstrates the complexity of interactions in a full size climate model, which leads to similar complex response patterns on small parameter changes. The same diagram for PUMA figure 2.1 (right) shows the direct paths, which allow the easy identification of effects.

2.4 Requirements

PUMA is free of licenses, anyone may download and use it. But keep in mind, that it is not a toy-model. The design of experiments and the interpretation of the results requires a thorough knowledge of atmospheric sciences.

PUMA is available as FORTRAN-90 source code. So all, that is needed to use PUMA on any computer is a FORTRAN-90 compiler. The GUI requires additionally a C-compiler and the graphical library X11, which is standard on any UNIX/Linux system and available for other operating systems like MAC-OS and Windows.

The program was developed and tested with several operating systems like LINUX, Solaris, Windows, and Unicos. Main development is done with Linux and the Intel-FORTRAN-90 compiler which is freely available for non-commercial usage.

The postprocessor Pumaburner requires a C++ compiler.

There are several compilers available for the Linux operating system. Most, PUMA, and Planet Simulator was successfully tested with:

- SunStudio12 (development environment including FORTRAN-90, C, C++, and Debugger) for Solaris and Linux. SunStudio12 can be downloaded for free from `http://www.sun.com`.
- Intel FORTRAN-90 compiler. There is a cost-free version of the Intel compiler for non-commercial use `http://www.intel.com`.
- Gnu FORTRAN (gfortran). This free and open FORTRAN-90 compiler is part of some Linux distributions (e.g. Fedora 7). It's also available from `http://directory.fsf.org/devel/compilers/gfortran.html`.

2.5 History

The PUMA version at the University of Hamburg originates from the Hoskins and Simmons SGCM (Simple General Circulation Model) version (Hoskins and Simmons, 1975). The major differences between PUMA and its predecessor SGCM are:

- The code is rewritten in portable FORTRAN-90 code, which removes problems associated with machine-specific properties like word lengths, floating point precision, output, etc. All necessary routines are in the source code, even the FFT (**F**ast **F**ourier **T**ransformation) and the Legendre transformation. The model can be run on any computer with a standard FORTRAN-90 compiler. The MPI-library is needed to run PUMA on parallel machines (see below). The Xlib (X11R6) library is needed for the graphical user interface.
- The truncation scheme is changed from jagged triangular truncation to standard triangular truncation that is compatible to other T-models like ECHAM.
- The PUMA/Pumaburner system is data compatible to ECHAM/Afterburner. Thus all other diagnostic software can be used on PUMA data.
- PUMA is fully parallelized and can use as many CPU's as latitudes (e.g. 32 in T21 resolution). It uses the MPI (**M**essage **P**assing **I**nterface library while running on parallel systems or cluster. MPI is not needed for running PUMA on a single CPU.

Chapter 3

Modules

This is a technical documentation of the PUMA model. In the following, the purposes of the individual modules is given and the general structure and possible input and output opportunities (namelist, files) are explained.

3.1 fftmod.f90 / fft991mod.f90

General The module fftmod.f90 contains all subroutines necessary to perform the fast fourier transformation and its inverse. The interface to the main PUMA module puma.f90 is given by the subroutines *gp2fc* and *fc2gp* which are called in puma.f90 from the subroutine *gridpoint*.

Input/Output fftmod.f90 does not use any extra input file or output file. No namelist input is required.

Structure Internally, fftmod.f90 uses the FORTRAN-90 module fftmod, which uses no other modules. Subroutine *gp2fc* performs the transformation from grid point space into fourier space while the subroutine *fc2gp* does the transformation from fourier space into grid point space. Both routines use several subroutines to do the direct or indirect transformation for different factors. When *gp2fc* or *fc2gp* is called the first time, *fftini* is called to do the initialization of the FFT.

The alternate module fft991mod.f90 may be used instead of fftmod.f90. While fftmod.f90 runs faster fft991mod.f90 can be used for resolutions, that are not supported by fftmod.f90, e.g. T63 or T106. Edit the file "Most15/puma/src/make_puma" for module selection. Use either

```
FFTMOD=fftmod
```

or

```
FFTMOD=fft991mod
```

3.2 fileseek.f90 / fseek64.f90 / fseek.f90

General The module fileseek.f90 is used by the restartmod.f90 for file positioning. Most compiler have FORTRAN callable routines **fseek** and **ftell** in their runtime libraries. The **gfortran** has **ftell**, but not **fseek**. In this case a subroutine, defined in fileseek.f90 is used instead. fseek64.f90 is a variant with a 64bit filepointer, that can address filepositions beyond 2 GBytes, though the PUMA restart files will never be that large. The configure script tries to figure out, which of these three variants to use and writes an entry to the file most_compiler.

Code This is a listing of the default fseek.f90:

```
! ====================
! SUBROUTINE FILESEEK
! ====================

! use this routine with FORTRAN compilers that support "fseek"

subroutine fileseek(kunit,kptr)
implicit none
integer :: kunit,kptr
call fseek(kunit,kptr,0)
return
end
```

3.3 guimod.f90 / guimod_stub.f90

General The module guimod.f90 contains subroutines for communication with the GUI. On operating systems, that don't support the Xlib (X11R6) (e.g. Windows), guimod_stub.f90 may be used as a stub replacement.

Structure The following subroutines are included in guimod.f90:

Subroutine	Purpose
guistart	initialize GUI
guistop	finalize GUI
guistep_puma	called every timestep from PUMA
guistep_plasim	called every timestep from PLASIM
guips	gather, scale, and send surface pressure to GUI
guihor	gather, scale, and send a gridpoint array to GUI
guigv	gather, scale, and send wind components to GUI
change_disp	called for user input in GUI
change_dtep	called for user input in GUI
change_dtns	called for user input in GUI
change_co2	called for user input in GUI
change_gsol0	called for user input in GUI
change_dawn	called for user input in GUI

3.4 legmod.f90

General The module legmod.f90 contains all subroutines necessary to perform the Legendre transformation and its inverse. The interface to the main PUMA module puma.f90 is given by the subroutines *legini*, *inigau*, *fc2sp*, *fc3sp*, and *sp2gp* which are called in puma.f90 from the subroutines *prolog* and *gridpoint*

Input/Output legmod.f90 does not use any extra input file or output file. No namelist input is required

The following subroutines are included in legmod.f90:

Subroutine	Purpose
inigau	compute Gaussian abscissas and weights
legini	compute Legendre polynomials
fs2sp	Fourier to Spectral transformation
sp2fc	Spectral to Fourier transformation
sp3fc	Simultaneous transformation of T, Div., and Vort.
mktend	Compute and transform tendencies

3.5 mpimod.f90 / mpimod_stub.f90

General The module mpimod.f90 contains interface subroutines to the MPI (Message Passing Interface) needed for (massive) parallel computing. Several MPI routines are called from the module. The interface to other modules are given by numerous subroutines which name starts with *mp*. Subroutines from mpimod.f90 are called in sveral other modules. There are no direct calls to MPI other than in mpimod.f90. This encapsulation makes it possible to use mpimod_stub.f90 for single CPU runs without changing any other part of the model code. The selection is done automatically by using MoSt or manually by editing "Most15/puma/src/make_puma".

Input/Output mpimod.f90 and mpimod_stub do not use any extra input file or output file. No namelist input is required

Structure Internally, mpimod.f90 uses the FORTRAN-90 module mpimod, which uses the global common module pumamod from pumamod.f90 and the MPI module mpi. mpimod_stub.f90 does not use any module. The following subroutines are included in mpimod.f90:

Subroutine	Purpose
mpbci	broadcast 1 integer
mpbcin	broadcast n integers
mpbcr	broadcast 1 real
mpbcrn	broadcast n reals
mpbcl	broadcast 1 logical
mpscin	scatter n integers
mpscrn	scatter n reals
mpscgp	scatter grid point field
mpgagp	gather grid point field
mpgallgp	gather grid point field to all
mpscsp	scatter spectral field
mpgasp	gather spectral field
mpgacs	gather cross section
mpgallsp	gather spectral field to all
mpsum	sum spectral field
mpsumsc	sum and scatter spectral field
mpsumr	sum n reals
mpsumbcr	sum and broadcast n reals
mpstart	initialize MPI
mpstop	finalize MPI

Subroutine	Purpose
mpreadgp	read and scatter grid point field
mpwritegp	gather and write grid point field
mpwritegph	gather and write (with header) grid point field
mpreadsp	read and scatter spectral field
mpwritesp	gather and write spectral field
mpi_info	give information about setup
mpgetsp	read spectral array from restart file
mpgetgp	read gridpoint array from restart file
mpputsp	write spectral array to restart file
mpputgp	write gridpoint array to restart file
mpmaxval	compute maximum value of an array
mpsumval	compute sum of all array elements

3.6 puma.f90

General The module puma.f90 is the main module of the model. It includes the main program *puma* and controls the run. From puma.f90 the interface routines to all other modules are called. The output is done by calling the subroutine to outsp. In addition, the adiabatic tendencies and the horizontal diffusion are computed in puma.f90. To do the necessary transformations, calls to the modules fftmod.f90 and legmod.f90 are used.

Input/Output puma.f90 A diagnostic printout is written on standard output (usuallay redirected with the operator ">" to a file). puma.f90 is controlled by the namelist *inp* which is part of the namelist file **puma_namelist**: For a complete list of namelist variables see appendix. Here is a table of the most important ones:

Parameter	Type	Purpose	Default
MPSTEP	Integer	MPSTEP (Minutes Per STEP) defines the length of the timestep. Recommended values are 60 min for T21 and 20 min for T42. The values are not checked. So take care not to violate the CFL (Courant-Friedrichs-Levy) critetion!	60
NYEARS	Integer	Number of years to be run	1
NMONTHS	Integer	Number of months to be run : NYEARS and NMONTHS may be used together. The simulation length in days is: NYEARS * 360 + NMONTHS * 30.	0
NOUTPUT	Integer	NOUTPUT is a global switch for enabling (1) or disabling (0) writing to **puma_output**.	
NWPD	Integer	NWPD (Number of Writes Per Day) defines the output interval for writing model arrays to the file **puma_output**. Possible values range from 1 (daily output) to 24 (hourly).	
NDIAG	Integer	NDIAG sets the interval (in timesteps) for printing out some diagnostic arrays and values to the standard output.	

Parameter	Type	Purpose	Default
NDL(NLEV)	Integer Array	Switch for diagnostic print out of a level (0 = off; 1 = on)	NLEV · 0
DTEP	Real	Equator to pole temperature difference [K] for Newtonian cooling	60.0
DTNS	Real	North to South pole temperature difference [K] for Newtonian cooling	0.0
DTROP	Real	Tropopause height [m] for Newtonian cooling	12000.0
DTTRP	Real	Smoothing of the tropopause [K] for Newtonian cooling	2
TGR	Real	Surface temperature [K] for Newtonian cooling	288
TDISS	Real	time scale [d] for the horizontal diffusion	0.25
PSURF	Real	Global mean sea level pressure [Pa]	101100.00
RESTIM(NLEV)	Real Array	Time scale [d] for Newtonian cooling	0.0
T0K(NLEV)	Real Array	Reference temperature used in the discretization scheme	250.0
TFRC(NLEV)	Real Array	Time scale [d] for Rayleigh friction (0.0 = off)	0.0

Structure After starting MPI, the main program *puma* calls *prolog* for initializing the model. Then, *master* is called to do the time stepping. Finally, subroutine *epilog* finishes the run. In subroutine *prolog*, calls to different subroutines, which are part of puma.f90 or are provided by other modules, initialize various parts of the model: *gauaw* and *inilat* build the grid, *readnl* reads the namelist and sets some parameter according to the namelist input, *initpm* and *initsi* initialize some parameter for the physics and the semi implicit scheme, respectively. *outini* starts the output. The prgram then checks for the existence of a file named "puma_restart". If the file can be opened then the restart record is read by *restart*, otherwise *initfd* sets the prognostic variables to their initial values. Finally, the global averaged surface pressure is set according to PSURF and the orography. Subroutine *master* controls the time stepping. First, if its not a restart, initial NKITS explicit forward timesteps are performed. The main loop is defined by calling *gridpoint* for the adiabatic tendencies, and *spectral* to add the adiabatic tendencies. interface routines to the output module outmod.f90. The run is finalized by subroutine *epilog* which writes the restart records and finalizes MPI.

3.7 pumamod.f90

General The module pumamod.f90 contains all parameters and variables which may be used to share information between puma.f90 and other modules. No subroutines or programs are included.

Input/Output pumamod.f90 does not use any extra input file or output file. No namelist input is required

Structure Internally, pumamod.f90 is a FORTRAN-90 module named pumamod. Names for global parameters, scalars and arrays are declared and, if possible, values are preset.

3.8 resmod.f90

General The module resmod.f90 defines horizontal and vertical resolution together with the number of processes.

Creation This file is created by the model starter MoSt. If you don't use MoSt, copy the file resmod_def.f90 to resmod.f90 and edit the file resmod.f90 to fit your needs.

Structure The default values are set to T21 (NLAT_ATM = 32) and 10 level (NLEV_ATM = 10). The number of processes to use is set to 1 (no parallel processing).

```
module resmod ! will be overwritten by MoSt

parameter(NLAT_ATM = 32)
parameter(NLEV_ATM = 10)
parameter(NPRO_ATM =  1)
end module resmod
```

All other resolution dependent parameters like NTRU, NLON, NHOR, etc. are derived from the values set in resmod.f90.

3.9 restartmod.f90

General The module restartmod.f90 contains routines for opening, reading and writing restart files. The scalars and array of the restart files are identified by name. This enables adding or removing variables from the restart file without loosing compatibility. There's also no dependence on the sequence of variables. In parallel runs these routines are either called from the root process, which takes care of broadcasting or from subroutines of the mpimod.f90 which gather before writing or scatter after reading arrays.

Structure

Subroutine	Purpose
restart_ini	Scan restart file and store pointer
restart_prepare	Open file for restart ouput
restart_stop	Close files
get_restart_integer	read integer scalar
get_restart_array	read real array
put_restart_integer	write integer scalar
put_restart_array	write real array

Chapter 4

Parallel Program Execution

4.1 Concept

PUMA is coded for parallel execution on computers with multiple CPU's or networked machines. The implementation uses MPI (Message Passage Interface), that is available for nearly every operating system `http://www.mcs.anl.gov/mpi`.

In order to avoid maintaining two sets of source code for the parallel and the single CPU version, all calls to the MPI routines are encapsulated into a module. Users, that want to compile and execute the parallel version use the module **mpimod.f90** and the commands **mpif90** for compiling and **mpirun** for running.

If MPI is not implemented or the single CPU version is sufficient, the module **mpimod_dummy.f90** is used instead of mpimod.f90. Also remove or comment the line:

```
!     use mpi
```

and set the number of processors to 1:

```
      parameter(NPRO = 1)
```

4.2 Parallelization in Gridpoint Domain

The data arrays in gridpoint domain are either three-dimensional e.g. gt(NLON, NLAT, NLEV) referring to an array organized after longitudes, latitudes and levels, or two-dimensional, e.g. gp(NLON, NLAT). The code is organized such, that there are no dependencies in latitudinal direction, while in gridpoint domain. Such dependencies are resolved during the Legendre-Transformations. So the the partitioning of the data is done in latitudes. The program can use as many CPU's as latitudes with the extreme of every CPU doing the computations for a single latitude. There is the restriction however, that the number of latitudes (NLAT) divided by the number of processors (NPRO), giving the number of latitudes per process (NLPP) must have zero remainder. E.g. A T31 resolution uses $NLAT = 48$. Possible values for NPRO are then 1, 2, 3, 4, 6, 8, 12, 16, 24, and 48.

All loops dealing with a latitudinal index look like:

```
      do jlat = 1 , NLPP
         ....
      enddo
```

There are, however, many subroutines, with the most prominent called **calcgp**, that can fuse latitudinal and longitudinal indices. In all these cases the dimension NHOR is used. NHOR is defined as: $NHOR = NLON * NLPP$ in the pumamod - module. The typical gridpoint loop that looks like:

```
do jlat = 1 , NLPP
   do jlon = 1 , NLON
      gp(jlon,jlat) = ...
   enddo
enddo
```

is then replaced by the faster executing loop:

```
do jhor = 1 , NHOR
   gp(jhor) = ...
enddo
```

4.3 Parallelization in Spectral Domain

The number of coefficients in spectral domain (NRSP) is divided by the number of processes (NPRO) giving the number of coefficients per process (NSPP). The number is rounded up to the next integer and the last process may get some additional dummy elements, if there is a remainder in the division operation.

All loops in spectral domain are organized like:

```
do jsp = 1 , NSPP
   sp(jsp) = ...
enddo
```

4.4 Synchronization points

All processes must communicate and have therefore to be synchronized at following events:

- Legendre-Transformation: This involves changing from latitudinal partitioning to spectral partitioning and such some gather and scatter operations.
- Inverse Legendre-Transformation: The partitioning changes from spectral to latitudinal by using gather, broadcast, and scatter operations.
- Input-Output: All read and write operations must be done only by the root process, who gathers and broadcasts or scatters the information as desired. Code that is to be executed by the root process exclusively is written like:

```
if (mypid == NROOT) then
   ...
endif
```

NROOT is typically 0 in MPI implementations, mypid (My process identification) is assigned by MPI.

4.5 Source code

It needs some discipline in order to maintain parallel code. Here are the most important rules for changing or adding code to **PUMA**:

- Adding namelist parameters: All namelist parameters must be broadcasted after reading the namelist. (Subroutines mpbci, mpbcr, mpbcin, mpbcrn)
- Adding scalar variables and arrays: Global variables must be defined in a module header and initialized.
- Initialization code: Initialization code, that contains dependencies on latitude or spectral modes must be done by the root process only and then scattered from there to all child processes.
- Array dimensions and loop limits: Always use parameter constants (NHOR, NLAT, NLEV, etc.) as defined in pumamod.f90 for array dimensions and loop limits.
- Testing: After significant code changes the program should be tested in single and in multi-CPU configuration. The results of a single CPU run is usually not exactly the same as the result of a multi-CPU run due to effects in rounding. But the results should show only small differences during the first timesteps.
- Synchronization points: The code is optimzed for parallel execution and minimizes therefore communication overhead. The necessary communication code is grouped around the Legendre-transformations. If more scatter/gather operations or other communication routines are to be added, they should be placed just before or after the execution of the calls to the Legendre-Transformation. Any other place would degrade the overall performance in introducing additional process synchronization.

Chapter 5

Graphical User Interface

5.1 Graphical user interface (GUI)

PUMA may be used in the traditional fashion, with shell scripts, batch jobs, and network queuing systems. This is acceptable for long running simulations on complex machines and number-crunchers, like vector- computers, massive-parallel-computers and workstation clusters. There is now, however, a much more convenient method by using a graphical user interface (GUI) for model setup with parameter configurations and for interaction between user and model.

PUMA is configured and setup by the first GUI module named MoSt (Model Starter, screenshot in 5.1). MoSt is the fastest way to get the model running. It gives access to the most important parameters of the model preset to the most frequently used values. The model can be started with a mouse click on the button labelled "Save & Run" either with the standard paramater setting or after editing some of the parameters in the MoSt window. Some parameters, like horizontal and vertical resolution, or the number of processors, require the building (compile, link and load) of new executables. MoSt achieves this by generating and executing build scripts, that perform the necessary code changes and create the required executable. Other parameters define startup- and boundary conditions or settings for parameterisations. They can be edited in MoSt and, after a check for correct range and consistency with other parameters, are written to the model's namelist file.

Depending on all settings MoSt generates a runscript for the simulation. The user has the choice of leaving MoSt and continue with the simulation under control of a GUI right away, or to exit MoSt with the scripts prepared to run. The second alternative is useful for users, who want to modify the setup beyond the scope of MoSt or want to run the Planet Simulator without GUI.

There's also a simple graphical editor for topograpy. Check the box Orography and then use the mouse to mark rectangular areas in the topography display. Enter a value for rising (positive) or lowering the area and press the button labelled **Preprocess**. The preprocessor will be built and executed, a new topography will be computed and written to a start file.

Another editor is the mode editor for spherical harmonics. Green modes are enabled, red modes are disabled. This feature can be used to make runs with only certain modes of spherical harmonics being active. MB1, MB2, MB3 refer to the left, middle, and right mouse button. You may toggle individual modes or whole lines and columns. Currently this mode editor can only be used for PUMA in the T21 resolution.

The GUI for running PUMA (screenshot in 5.2) has two main purposes. The first one is to display model arrays in suitable representations. Current implementations are:

- Zonal mean cross sections

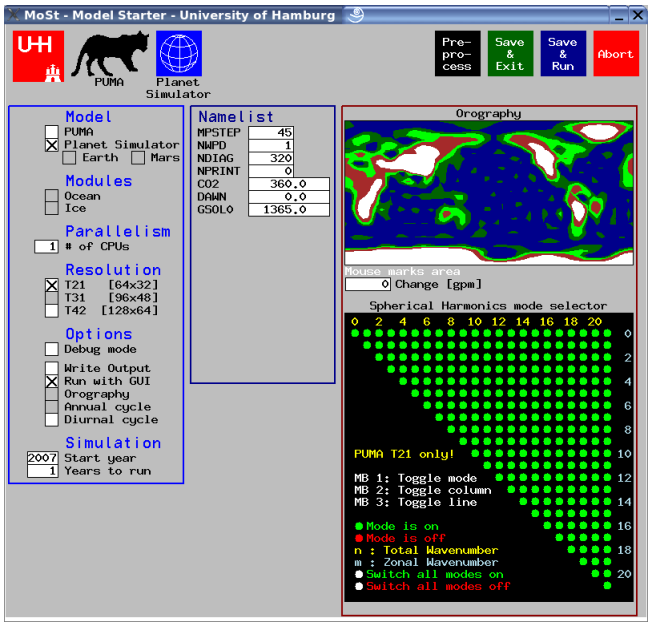


Figure 5.1: Screenshot of Model Starter (MoSt)

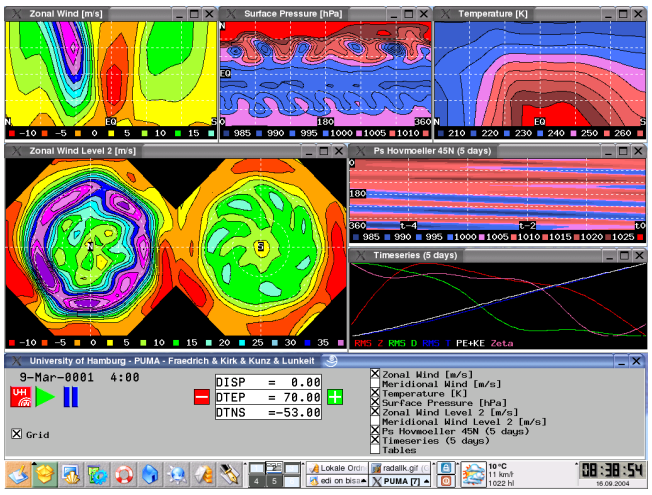


Figure 5.2: Screenshot of Graphical User Interface (GUI)

- Horizontal global fields in cylinder projection
- Horizontal global fields in polar projection
- Time-longitude (Hovmoeller) diagrams
- Amplitudes of coefficients of spherical harmonics
- Time series
- Numerical values

In case of horizontal global grids pressing the MMB (Middle Mouse Button) toggles between cylinder and polar projection. If the grid is just one level from many of a three dimensional field like u or v, the level shown can be decreased by the LMB or increased by the RMB. For Hovmoeller and longitude height sections the LMB and RMB can be used to select the latitude.

The second purpose is the interaction part of the GUI, which allows the user to change selected model variables during the model run. It is not necessary, though possible, to pause the model while changing variables. Changes to model variables are, of course, monitored in the outputfile and checked by GUI for the appropriate range of values and maximum possible change per timestep because, otherwise, a rapid parameter change or a choice of values beyond the normal range may blow up the model.

All model variables, which are candidates for the display or interactive changes, have a special code to communicate with the Planet Simulator. The experienced modeller can add new code for more variables using the existing communication code as template. Thus all model fields or even fields received via coupling with other models can be put on the GUI display.

Both, MoSt and GUI are implemented using the Xlib (X11R5), which is a library of routines for graphics and event communication. As this library is part of every UNIX/Linux operating system and base of all desktop environments, there is no need to install additional software for running MoSt and GUI. Another important property of Xlib is the full network transparency. The display of MoSt and GUI is not locked to the machine running the programs or the model. In fact, the best performance is obtained in running the Planet Simulator on two or four CPUs of a remote server while displaying the GUI on the user's workstation. In summarizing, the MoSt and GUI programs automate many tedious tasks, minimize the time to become familiar with the Planet Simulator, and make debugging and parameter tuning much easier. More kinds of presentations, coordinate projections and interactivity are being developed. A graphical preprocessor with editor for boundary conditions and a graphical postprocessor are future expansions to build an almost complete environment for modellers.

5.2 GUI configuration

On initialization the GUI reads its configuration from a file **GUI.cfg** which must be present in the current directory. MoSt copies the file **GUI.cfg** from the ../dat/ directory to the run directory while building PUMA. After reading **GUI.cfg** an attempt is made to read the file **GUI_last_used.cfg**. This file is always written at the end of a GUI controlled simulation. So one may rearrange and position GUI windows during a run and the new layout will be saved to the file **GUI_last_used.cfg**. In order to make this user layout default for following runs, just copy this file like:

```
Most15/puma/run$ cp ../dat/GUI.cfg ../dat/GUI.cfg.old
Most15/puma/run$ cp GUI_last_used.cfg ../dat/GUI.cfg
```

MoSt will then copy your new layout to the run directory at the next invocation.

The **GUI.cfg** is a text file that may be also edited manually. There is a section for each window (counting from 0 to 8) which looks like:

```
[Window 00]                          <- window number (0..8)
Array:CSU                            <- array name
Plot:ISOCS                           <- picture type
Palette:U                            <- colour palette
Title:Zonal Wind [m/s]               <- window title
Geometry:  529  299    2    3       <- width height left top

[Window 01]
Array:SPAN
Plot:ISOSH
Palette:AMPLI
Title:Spherical Harmonics Ps
Geometry:  529  299  535    3

...
```

Possible values for these items are:

5.2.1 Array

Name	Description
CSU	Cross Section U - Zonal mean zonal wind
CSV	Cross Section V - Zonal mean meridional wind
CST	Cross Section T - Zonal mean temperature
SPAN	Spherical harmonics coefficients of surface pressure
GU	Three dimensional grid of zonal wind
GV	Three dimensional grid of meridional wind
GP	Grid of surface pressure
SCALAR	Selected scalars for timeseries and tables

5.2.2 Plot

Name	Description
ISOHOR	Isolines and colouring of horizontal grids
ISOCS	Isolines and colouring of cross sections
ISOHOV	Colouring of Hovmoeller diagram
ISOTS	Timeseries
ISOTAB	Tables
ISOSH	Coloured amplitudes
ISOLON	Isolines and colouring of longitude height section

5.2.3 Palette

Name	Range	Description
AUTO	automatic	rainbow colours
U	-10 .. 50	rainbow colours
V	-10 .. 10	rainbow colours
T	-50 .. 50	blue - red
P	985 .. 1025	blue - red
Q	0 .. 60	rainbow colours
MARST	-90 .. 0	blue -red
AMPLI	0 .. 12	blue - green -red
VEG	0 .. 100	shades of green

5.2.4 Title

The title item may contain any text, but keep it short, the length of the window's title bar is limited. The words Latitude and Level have special features in conjunction with threedimensional arrays, where the user may scroll the level or latitude. The GUI will insert the level number after the world Level or the latitude after the word Latitude.

5.2.5 Geometry

The four integers following the geometry item describe the size and screen position of the window. The first two parameters refer to width and height in screen pixel. These are the sizes of the inner window, title bar, border and other decorations are not counted. The third and fourth parameter set the coordinates of the upper left corner of the window x and y, again without borders. If the geometry item is not defined, the GUI will initialize the window's geometry depending on the screen size.

Chapter 6

Postprocessor Pumaburner

6.1 Introduction

The **Pumaburner** is a postprocessor for the **Planet Simulator** and the **PUMA** model family. It's the only interface between *raw* model data output and diagnostics, graphics, and user software.

The output data of the **Planet Simulator** are stored as packed binary (16 bit) values using the model representation. Prognostic variables like temperature, divergence, vorticity, pressure, and humidity are stored as coefficients of spherical harmonics on σ levels. Variables like radiation, precipitation, evaporation, clouds, and other fields of the parameterization package are stored on Gaussian grids.

The tasks of the **Pumaburner** are:

- Unpack the *raw* data to full real representation.
- Transform variables from the model's representation to a user selectable format, e.g. grids, zonal mean cross sections, fourier coefficients.
- Calculate diagnostic variables, like vertical velocity, geopotential height, wind components, etc.
- Transfrom variables from σ levels to user selectable pressure levels.
- Compute monthly means and standard deviations.
- Write selected data either in SERVICE or NetCDF format for further processing.

6.2 Usage

```
burn5 [options] InputFile OutputFile <namelist >printout
     option -h : help (this output)
     option -c : print available codes and names
     option -d : debug mode (verbose output)
     option -n : NetCDF output (override namelist option)
     option -m : Mean=1 output (override namelist option)
     InputFile : Planet Simulator or PUMA data file
    OutputFile : SERVICE or NetCDF format file
      namelist : redirected <stdin>
      printout : redirected <stdout>
```

6.3 Namelist

The namelist values control the selection, coordinate system and output format of the post-processed variables. Names and values are not case sensitive. You can assign values to the following names:

Name	Def.	Type	Description	Example
HTYPE	S	char	Horizontal type	HTYPE=G
VTYPE	S	char	Vertical type	VTYPE=P
MODLEV	0	int	Model levels	MODLEV=2,3,4
hPa	0	real	Pressure levels	hPa=500,1000
LATS	0	int	No. of latitudes for output grid	LATS=40
LONS	0	int	No. of longitudes for output grid	LONS=80
CODE	0	int	ECMWF field code	CODE=130,152
NETCDF	0	int	NetCDF output selector	NETCDF=1
CYCLICAL	0	int	Add data for longitude=360	CYCLICAL=0
MEAN	1	int	Compute monthly means	MEAN=0
HHMM	1	int	Time format in Service format	HHMM=0
HEAD7	0	int	User parameter	HEAD7=0815
MARS	0	int	Use constants for planet Mars	MARS=1
MULTI	0	int	Process multiple input files	MULTI=12

6.4 HTYPE

HTYPE accepts the first character of the following string. Following settings are equivalent: HTYPE = S, HTYPE=Spherical Harmonics HTYPE = Something. Blanks and the equal-sign are optional.
Possible Values are:

Setting	Description	Dimension for T21 resolution
HTYPE = S	Spherical Harmonics	(506):(22 * 23 coefficients)
HTYPE = F	Fourier Coefficients	(32,42):(latitudes,wavenumber)
HTYPE = Z	Zonal Means	(32,levels):(latitudes,levels)
HTYPE = G	Gauss Grid	(64,32):(longitudes,latitudes)

6.5 VTYPE

VTYPE accepts the first character of the following string. Following settings are equivalent: VTYPE = S, VTYPE=Sigma VTYPE = Super. Blanks and the equal-sign are optional.
Possible Values are:

Setting	Description	Remark
VTYPE = S	Sigma (model) levels	Some derived variables are not available
VTYPE = P	Pressure levels	Interpolation to pressure levels

6.6 MODLEV

MODLEV is used in combination with **VTYPE = S**. If VTYPE is not set to Sigma, the contents of MODLEV are ignored. MODLEV is an integer array that can get as many values as there are levels in the model output. The levels are numbered from top of the atmosphere to the bottom. The number of levels and the corresponding sigma values are listed in the pumaburner printout. The outputfile orders the level according to the MODLEV values.

MODLEV=1,2,3,4,5 produces an output file of five model levels sorted from top to bottom, while MODLEV=5,4,3,2,1 sorts them from bottom to top.

6.7 hPa

hPa is used in combination with **VTYPE = P**. If VTYPE is not set to Pressure, the contents of hPa are ignored. hPa is a real array that accepts pressure values with the units hectoPascal or millibar. All output variables will be interpolated to the selected pressure levels. There is no extrapolation on the top of the atmosphere. For pressure values, that are lower than that of the model's top level, the top level value of the variable is taken. The variables temperature and geopotential height are extrapolated if the selected pressure is higher than the surface pressure. All other variables are set to the value of the lowest mode level for this case. The outputfile contains the levels in the same order as set in hPa. Example: hpa = 100,300,500,700,850,900,1000.

6.8 LATS and LONS

The pumaburner defaults to the dimension of the model run. E.g. Lats=32 and Lons=64 for a T21 resolution. Note however, that this results in Gaussian grids with non equidistant latitudes. Selecting for Lats and Lons values, that are different from the internal resolution produces equidistant lat-lon grids. Lats sets the number of latitudes from North to South, with the northpole at index 1 and the southpole at index Lats. Delta Phi is therefore 180 degrees / (Lats - 1). Lons sets the number of gridpoints on every latitude circle. Delta Lambda is 360 / Lons. Index 1 is on Greewich (0 degrees), while the last index denotes the point (360 degrees - Delta Lambda). Technical note: Variables, that are stored as spherical harmonics (Temperature, vorticity, divergence, etc.) are calculated on the user grid by setting up the Legendre transformation and the FFT accordingly. Variables, that are stored on Gaussian grids are interpolated with a bilinear interpolation. Note: Lats ¿= 8 and Lons ¿= 16 due to technical reasons.

6.9 MEAN

MEAN can be used to compute montly means and/or deviations. The Pumaburner reads date and time information from the model file and handles different lengths of months and output intervals correctly.

Setting	Description
MEAN = 0	Do no averaging - all terms are processed.
MEAN = 1	Compute and write monthly mean fields. Not for spherical harmonics, Fourier coefficients or zonal means on sigma levels.
MEAN = 2	Compute and write monthly deviations. Not for spherical harmonics, Fourier coefficients or zonal means on sigma levels. Deviations are not available for NetCDF output.
MEAN = 3	Combination of MEAN=1 and MEAN=2. Each mean field is followed by a deviation field with an identical header record. Not for spherical harmonics, Fourier coefficients or zonal means on sigma levels.

6.10 Format of output data

The **pumaburner** supports three different output formats:

- **NetCDF** (Network Common Data Format)
- **Service** Format for user readable data (see below).

For more detailed descriptions see for example:
`http://www.nws.noaa.gov/om/ord/iob/NOAAPORT/resources/`

Setting	Description
NetCDF = 1	The output file is written in NetCDF format.
NetCDF = 0	The output file is written in Service format.

6.11 SERVICE format

The SERVICE format uses the following structure: The whole file consists of pairs of header records and data records. The header record is an integer array of 8 elements.

```
    head(1) = ECMWF field code
    head(2) = modellevel or pressure in [Pa]
    head(3) = date  [yymmdd]  (yymm00 for monthly means)
    head(4) = time  [hhmm]  or [hh] for HHMM=0
    head(5) = 1. dimension of data array
    head(6) = 2. dimension of data array
    head(7) = may be set with the parameter HEAD7
    head(8) = experiment number (extracted from filename)

    Example for reading the SERVICE format (NETCDF=0)

    INTEGER HEAD(8)
    REAL    FIELD(64,32)     ! dimensions for T21 grids
    READ (10,ERR=888,END=999) HEAD
    READ (10,ERR=888,END=999) FIELD
    ....
888 STOP 'I/O ERR'
999 STOP 'EOF'
    ....
```

6.12 HHMM

Setting	Description
HHMM = 0	head(4) shows the time in hours (HH).
HHMM = 1	head(4) shows the time in hours and minutes (HHMM).

6.13 HEAD7

The 7th. element of the header is reserved for the user. It may be used for experiment numbers, flags or anything else. Setting HEAD7 to a number exports this number to every header record in the output file (SERVICE format only).

6.14 MARS

This parameter is used for processing simulations of the Mars atmosphere. Setting MARS=1 switches gravity, gas constant and planet radius to the correct values for the planet Mars.

6.15 MULTI

The parameter MULTI can bes used to process a series of input data within one run of the pumaburner. Setting MULTI to a number (n) tells the pumaburner to procees (n) input files. The input files must follow one of the following two rules:

- YYMM rule: The last four characters of the filename contain the data in the form YYMM.
- .NNN rule: The last four characters of the filename consist of a dot followed ny a 3-digit sequence number.

```
Examples:

Namelist contains MULTI=3
Command: pumaburn <namelist >printout run.005 out
pumaburn processes the files <run.005> <run.006> <run.007>

Namelist contains MULTI=4
Command: pumaburn <namelist >printout exp0211 out
pumaburn processes the files <exp0211> <exp0212> <exp0301> <exp0302>
```

6.16 Namelist example

```
VTYPE  = Pressure
HTYPE  = Grid
CODE   = 130,131,132
hPa    = 200,500,700,850,1000
MEAN   = 0
NETCDF = 0
```

This namelist will write Temperature(130), u(130) and v(131) on pressure levels 200hPa, 500hPa, 700hPa, 850hPa and 1000hPa. The output interval is the same as found on the model data, e.g. every 12 or every 6 hours (MEAN=0). The output format is SERVICE format.

6.17 Troubleshooting

If the pumaburner reports an error or doesn't produce the expected results, try the following:

- Check your namelist, especially for invalid codes, types and levels.
- Run the pumaburner in debug-mode by using the option -d. Example:

  ```
  pumaburn <namelist >printout -d data.in data.out
  ```

  This will print out some details like parameters and memory allocation during the run. The additional information may help to detect the problem.

- Not all combinations of HTYPE, VTYPE, and CODE are valid. Try to use HTYPE=Grid and VTYPE=Pressure before switching to exotic parameter combinations.

Chapter 7

Graphics

7.1 Grads

In this section, visualisation using the graphics package GrADS is described. A useful Internet site for reference and installation instructions is

<http://grads.iges.org/grads/grads.html>.

Latest versions of GrADS can handle data in NETCDF format via the command sdfopen. Any file produced by the pumaburner with the option NETCDF=1 can be read directly from GrADS. An alternate way for files in SERVICE format is to use a converter, which translates the SERVICE format to NETCDF. In the following it is assumed that the PUMA output has been converted to SERVICE format with the pumaburner and the resulting file is called puma.srv. Monthly mean data is either obtained directly from the pumaburner (namelist parameter MEAN=1, see section 6) or via a PINGO command:

```
srv monmeans puma.srv puma_m.srv
```

Information on the PINGO package can be found in DKRZ report 11 at

<http://www.mad.zmaw.de/Pingo/repdl.html>.

The SERVICE file has to be converted to GrADS's native format by the command:

```
srv2gra puma_m.srv
```

which results in the files puma_m.gra and puma_m.ctl. The first file contains the data, the latter one information on the grid, time steps, and variable names. The program srv2gra is one of the postprocessing tools available at

<http://puma.dkrz.de/puma/download/map/>.

If you chose to compile it yourself, please read the comments in the first few lines of the program text. Sometimes the srv2gra tool has difficulties to calculate an appropriate time increment from the date headers of the data records, so you should check this. In this example the file puma_m.ctl should look like this:

```
DSET ^puma_m.gra
UNDEF 9e+09
XDEF     64 LINEAR   0.0000   5.6250
OPTIONS YREV
YDEF     32 LEVELS
```

```
  -85.7606  -80.2688  -74.7445  -69.2130  -63.6786  -58.1430  -52.6065  -47.0696
  -41.5325  -35.9951  -30.4576  -24.9199  -19.3822  -13.8445   -8.3067   -2.7689
    2.7689    8.3067   13.8445   19.3822   24.9199   30.4576   35.9951   41.5325
   47.0696   52.6065   58.1430   63.6786   69.2130   74.7445   80.2688   85.7606
ZDEF  1 LINEAR 1 1
TDEF 12 LINEAR 00:00Z01jan0001        1mo
VARS  3
c139     0 99    139     0     0
c151     0 99    151     0     0
c175     0 99    175     0     0
ENDVARS
```

Here, the line starting with `TDEF` ends with `1mo`, since we are handling monthly mean data. When the `PUMA` output is used without averaging, this should correspond to the output interval given by the `nafter` variable used in the `namelist` of your `PUMA` run (see section C). The number of variables depends on how the pumaburner was called. In this example, only 3 variables were processed, i.e. the surface temperature (`c139`), the sea level pressure (`c151`) and the albedo (`c175`; refer to appendix B for a list of codes).
The GrADS program is started by typing `grads` in a terminal window. Then, data is visualised either by typing commands line-by-line, or, preferably, by using scripts. The following script, called `tglob.gs`, displays the monthly mean surface temperature:

```
# tglob.gs
function pass(m)
'reinit'
'open puma_m'
'enable print print.mf'
'set t 'm
'c'
'set gxout shaded'
'd (c139-273.16)'
'cbar.gs'
'set gxout contour'
'd (c139-273.16)'
'draw title Surface Temperature (deg C) month 'm
'print'
'disable print'
'!gxps -i print.mf -o tglob'm'.ps'
```

The variable `m` at the beginning of the script defines the month which should be displayed. It is passed from the terminal with the script call. Note that in this line, no quotation marks are present, since only `GrADS` specific commands are framed by quotation marks. Script commands, like variable definitions, if-clauses etc. are used without quotation marks. The script is executed by typing its name without the ending and the number of the month to be shown. For example, `tglob 7` displays the monthly mean surface temperature in July. The resulting output file is called `tglob7.ps`.

The following script `thh` displays the time dependent surface temperature of Hamburg. Here, two variables are passed to `GrADS`, the first and last day to plot (note that here, the file `puma.gra` is opened, which contains data on a daily basis). The call `thh 91 180` displays the surface temperature of Hamburg for the spring season from April 1st to June 30th.

```
# thh.gs
function pass(d1 d2)
'reinit'
'open puma'
'enable print print.mf'
'set lat 53'
'set lon 10'
'set t 'd1' 'd2
'c'
'd (c139-273.16)'
'draw title Surface Temperature (deg C) in Hamburg'
'print'
'disable print'
'!gxps -i print.mf -o thh.ps'
```

It is possible to have more than one figure in a plot, which is illustrated in the following script. It plots seasonal means of the sea level pressure. The data file is prepared like this:

```
srv selcode,151 puma.srv slp.srv
srv seasmean slp.srv slp_sm.srv
srv2gra slp_sm.srv
```

The commands `set vpage` sets virtual pages inside the graphic window. The full window is 11 inch wide and 8.5 inch high, so `set vpage 0 5.5 4.25 8.5` defines the upper left corner. If `setlevs=1` is specified, the pressure levels as given are used. Otherwise, `GrADS` defines contour levels depending on the data set.

```
# slp_sm.gs
setlevs=1
'reinit'
'open slp_sm'
'enable print print.mf'
'c'
'set vpage 0 5.5 4.25 8.5'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
'set t 1'
'd c151/100'
'draw title SLP [hPa] yr 'ny' DJF'
'set vpage 5.5 11 4.25 8.5'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
```

```
'set t 2'
'd c151/100'
'draw title yr 'ny' MAM'
'set vpage 0 5.5 0 4.25'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
'set t 3'
'd c151/100'
'draw title yr 'ny' JJA'
'set vpage 5.5 11 0 4.25'
'set gxout contour'
if (setlevs=1)
'set clevs 990 995 1000 1005 1010 1015 1020'
endif
'set ccols 1'
'set grads off'
'set t 4'
'd c151/100'
'draw title yr 'ny' SON'
'print'
'disable print'
'!gxps -c -i print.mf -o slp_sm.ps'
```

7.2 Vis5D

> *"Vis5D is a system for interactive visualization of large 5-D gridded data sets such as those produced by numerical weather models. One can make isosurfaces, contour line slices, colored slices, volume renderings, etc of data in a 3-D grid, then rotate and animate the images in real time. There's also a feature for wind trajectory tracing, a way to make text annotations for publications, support for interactive data analysis, etc."*
>
> from the Vis5D home page,
> `http://www.ssec.wisc.edu/~billh/vis5d.html`

This powerful visualisation tool together with its documentation is available through the above home page. Vis5D uses its own data format which makes it necessary to transform your data. Depending on their format and the flowchart on `http://puma.dkrz.de/puma/download/map/` you have the following choices: If

- *your data is raw* `PUMA` *output,*
  you need to process it with the `pumaburner` postprocessor (see section 6) in order to transform it to either `NETCDF` (option `-n` or namelist parameter `NETCDF=1`) or `GRIB` (option `-g` or namelist parameter `GRIB=1`) and proceed from there.

- *your data is in* `SERVICE` *format,*
  you need to convert it to either `GRIB`, for instance with the `PINGO`s:

```
grb copy2 data.srv data_with_grib_metainfo.grb output.grb,
```

or NETCDF, using the program puma2cdf, which is available with the PUMA postprocessing tools. Despite of its name this program cannot process raw PUMA output but takes SERVICE format as input. It can as well be called as srv2cdf which changes its behaviour: oddities of model output such as the existence of February, 30^{th} are then no longer removed. Once the format is changed proceed from there.

- *your data is in* NETCDF *format,*
  it can easily transformed to Vis5D's native format by means of the program cdf2v2d, which is available with the PUMA postprocessing tools.

- *your data is in* GRIB *format,*
  you can find a transformation tool named Grib2V5d at
  <http://grib2v5d.sourceforge.net> which offers various practical features.

Once the conversion to Vis5D's native format is achieved please follow the instructions from the Vis5D documentation or, if Vis5D is already installed on your system, try finding your own way by typing:

```
vis5d my_data.v5d
```

Chapter 8

Model Dynamics

8.1 Model equations and numerics

Core of the model are the primitive equations. They describe the conservation of momentum, mass, and thermal energy. In spherical coordinates and in the sigma system and with the aid of the equation of state they can be written in the dimensionless form as follows:

Conservation of momentum:

Vorticity equation

$$\frac{\partial(\zeta + f)}{\partial t} = \frac{1}{(1-\mu^2)}\frac{\partial F_v}{\partial \lambda} - \frac{\partial F_u}{\partial \mu} + P_\zeta \tag{8.1}$$

Divergence equation

$$\frac{\partial D}{\partial t} = \frac{1}{(1-\mu^2)}\frac{\partial F_u}{\partial \lambda} + \frac{\partial F_v}{\partial \mu} - \nabla^2\left(\frac{U^2+V^2}{2(1-\mu^2)} + \Phi + T_0 \ln p_s\right) + P_D \tag{8.2}$$

Hydrostatic approximation

$$\frac{\partial \Phi}{\partial \ln \sigma} = -T \tag{8.3}$$

Conservation of mass:

Continuity equation

$$\frac{\partial \ln p_s}{\partial t} = -\int_0^1 A d\sigma \tag{8.4}$$

Conservation of energy:

First law of thermodynamics

$$\frac{\partial T'}{\partial t} = -\frac{1}{(1-\mu^2)}\frac{\partial (UT')}{\partial \lambda} - \frac{\partial (VT')}{\partial \mu} + DT' - \dot{\sigma}\frac{\partial T}{\partial \sigma} + \kappa\frac{T}{p}\omega + \frac{J}{c_p} + P_T, \tag{8.5}$$

with:

$$F_u = V(\zeta + f) - \dot{\sigma}\frac{\partial U}{\partial \sigma} - T'\frac{\partial \ln p_s}{\partial \lambda}$$

$$F_v = -U(\zeta + f) - \dot{\sigma}\frac{\partial V}{\partial \sigma} - T'(1-\mu^2)\frac{\partial \ln p_s}{\partial \mu}$$

$$A = D + \vec{V}\cdot\nabla \ln p_s$$

and $\quad U = u\cos\phi,\ V = v\cos\phi.$

Here the variables denote:

T	temperature
T_0	reference temperature
$T' = T - T_0$	temperature deviation from T_0
ζ	relative vorticity
D	divergence
p_s	surface pressure
p	pressure
Φ	geopotential
t	time
λ, ϕ	longitude, latitude
$\mu = \sin\phi$	
$\sigma = p/p_s$	sigma vertical coordinate
$\dot{\sigma} = d\sigma/dt$	vertical velocity in σ-system
$\omega = dp/dt$	vertical velocity in p-system
u, v	zonal, meridional component of horizontal velocity
$\vec{V}$	horizontal velocity with components U, V
f	Coriolis parameter
J	diabatic heating rate
c_p	specific heat of dry air at constant pressure
κ	adiabatic coefficient

The set of differential equations contains the four prognostic equations (8.1), (8.2), (8.4), and (8.5). Vorticity ζ and divergence D are scaled by angular velocity of the earth Ω, pressure p and p_s by the global mean surface pressure $P_s = 1011\,hPa$, temperatures T and T_0 by $a^2\Omega^2/R$, geopotential Φ by $a^2\Omega^2/g$, and time t by Ω^{-1}, where a is the radius of the earth, R the gas constant of dry air, and g the gravitational acceleration. For the parameterizations P_ζ, P_D and P_T see section 8.2. The model can be run with or without orography.

The horizontal representation of any model variable is given by a series of spherical harmonics. If Q is an arbitrary model variable, then its spectral representation has the form

$$Q(\lambda, \mu, t) = \sum_\gamma Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{8.6}$$

Here, Y_γ are the spherical harmonics, and Q_γ the corresponding complex amplitudes, where $\gamma = (n, m)$ designates the spectral modes ($n = 1, 2, 3, \ldots$: total wave number; $m = 0, \pm1, \pm2, \pm3, \ldots$: zonal wave number), with $|m| \leq n$ [Holton, 1992]. The latter condition follows from the triangular truncation in wave number space. The truncation is done at the total wave number n_T, which can be set to $n_T = 21, 31, 42, 85, 127, 170$, i.e. the model can be used with the T21,...,T170 spectral resolution. The vertical resolution is given by n_L equidistant σ-levels with the standard value $n_L = 5$. At the upper ($\sigma = 0$) and lower boundary ($\sigma = 1$) of the model domain the vertical velocity is set to zero ($\dot{\sigma} = 0$).

The linear contributions to the tendencies are calculated in the spectral domain, the nonlinear contributions in grid point space. Therefor, at every time step, the necessary model variables are transformed from spectral to grid point representation by Legendre transformation and Fast fourier transformation (FFT) and then the calculated tendencies are transformed back into the spectral domain, where the time step is carried out [for the transform method see Orszag, 1970, Eliasen et al., 1970]. Because of the semi-implicit time integration scheme [Hoskins and Simmons, 1975, Simmons, Hoskins, and Burridge, 1978] the terms due to gravity wave propagation are integrated in time implicitly, the remaining terms are integrated explicitly, the latter with a leap-frog time step. In the standard model version a time step of one hour is used. A Robert-Asselin time filter [Haltiner and Williams, 1982] is applied to avoid decoupling

of the two leap-frog time levels. The contributions to the tendencies due to vertical advection are calculated by an energy and angular-momentum conserving vertical finite-difference scheme [Simmons and Burridge, 1981].

8.2 Parameterizations

8.2.1 Friction

The dissipative processes in the atmosphere are parameterized by a linear approach (Rayleigh friction), describing the effect of surface drag and vertical transport of horizontal momentum due to small scale turbulence in the boundary layer. For this purpose, vorticity ζ and divergence D are damped towards the state of rest ($\zeta = 0$, $D = 0$) with the time scale τ_F.

The parameterization terms P_ζ and P_D appear in the model equations (8.1) resp. (8.2) and have the form:

$$P_\zeta = \frac{\zeta}{\tau_F} + H_\zeta \tag{8.7}$$

$$P_D = \frac{D}{\tau_F} + H_D. \tag{8.8}$$

The time scale $(\tau_F)_l$ depends on the σ-level l ($l = 1, \ldots, n_l$). Usually, for the upper levels ($l = 1, \ldots, n_l - 1$) it is set to $(\tau_F)_l = \infty$ (no friction) and for the lowest layer ($l = n_l$) a typical value is $(\tau_F)_l = 1\,d$. An explanation of the hyperdiffusion terms H_ζ and H_D follows in section 8.2.3.

8.2.2 Diabatic heating

All diabatic processes considered in the model are parameterized by a linear approach (Newtonian cooling), too, including the diabatic heating due to absorption and emission of short and longwave radiation as well as latent and sensible heat fluxes (convection). The temperature T relaxes towards the restoration temperature T_R with the time scale τ_R. The parameterization term in the thermal energy equation (8.5) is given by:

$$\frac{J}{c_p} + P_T = \frac{T_R - T}{\tau_R} + H_T. \tag{8.9}$$

For the hyperdiffusion H_T see section 8.2.3. τ_R depends on the σ-level l, T_R on the latitude ϕ and on the vertical coordinate σ. The restoration temperature field has the form

$$T_R(\phi, \sigma) = T_R(\sigma) + f(\sigma)\, T_R(\phi). \tag{8.10}$$

The vertical profile is described by

$$T_R(\sigma) = (T_R)_{tp} + \sqrt{\Big[\frac{L}{2}\big(z_{tp} - z(\sigma)\big)\Big]^2 + S^2} + \frac{L}{2}\big(z_{tp} - z(\sigma)\big), \tag{8.11}$$

with: $(T_R)_{tp} = (T_R)_{grd} - L\, z_{tp}$. Here, z denotes the geometric height, z_{tp} the global constant height of the tropopause, $L = -(\partial T_R)/(\partial z)$ the vertical restoration temperature gradient, $(T_R)_{grd}$ and $(T_R)_{tp}$ the restoration temperature at the surface and at the global isothermal tropopause, respectively. S provides a smoothing of the profile at the tropopause. $z(\sigma)$ is determined by an iterative method. The profile is determined by setting the parameters $(T_R)_{grd}$, z_{tp}, L and S. Figure 8.1 shows the vertical profile for the standard parameter values.

The temperature contrast between low and high latitudes due to the differential radiative energy balance, which drives the general circulation, is described by the meridional form of the restoration temperature:

$$T_R(\phi) = (\Delta T_R)_{NS} \frac{\sin\phi}{2} - (\Delta T_R)_{EP} \left(\sin^2\phi - \frac{1}{3}\right). \tag{8.12}$$

The meridional gradient decreases with height and vanishes at the tropopause:

$$f(\sigma) = \begin{cases} \sin\left(\frac{\pi}{2}\left(\frac{\sigma-\sigma_{tp}}{1-\sigma_{tp}}\right)\right) & \text{if} \quad \sigma \geq \sigma_{tp} \\ 0 & \text{if} \quad \sigma < \sigma_{tp}, \end{cases} \tag{8.13}$$

with the height of the tropopause

$$\sigma_{tp} = \left(\frac{(T_R)_{tp}}{(T_R)_{grd}}\right)^{\frac{g}{LR}}. \tag{8.14}$$

In equation (8.12), $(\Delta T_R)_{EP}$ represents the constant part of the meridional temperature contrast, $(\Delta T_R)_{NS}$ the variable part, corresponding to the annual cycle. Figure 8.2 shows the meridional and vertical form of the restoration temperature field (see eqn. (8.10)).

Usually, for the lower model levels, the time scale τ_R is set to less values (stronger diabatic heating) than for the upper levels to account for the stronger impact of the turbulent heat fluxes near the surface. A standard setting for τ_R is $(\tau_R)_{l=1,\ldots,3} = 30\,d$, $(\tau_R)_{l=4} = 10\,d$, $(\tau_R)_{l=5} = 5\,d$ for $n_l = 5$ levels.

8.2.3 Diffusion

The parameterizations (8.7), (8.8) and (8.9) further contain the hyperdiffusion terms H_ζ, H_D and H_T, respectively. The hyperdiffusion parameterizes both subgrid scale horizontal mixing

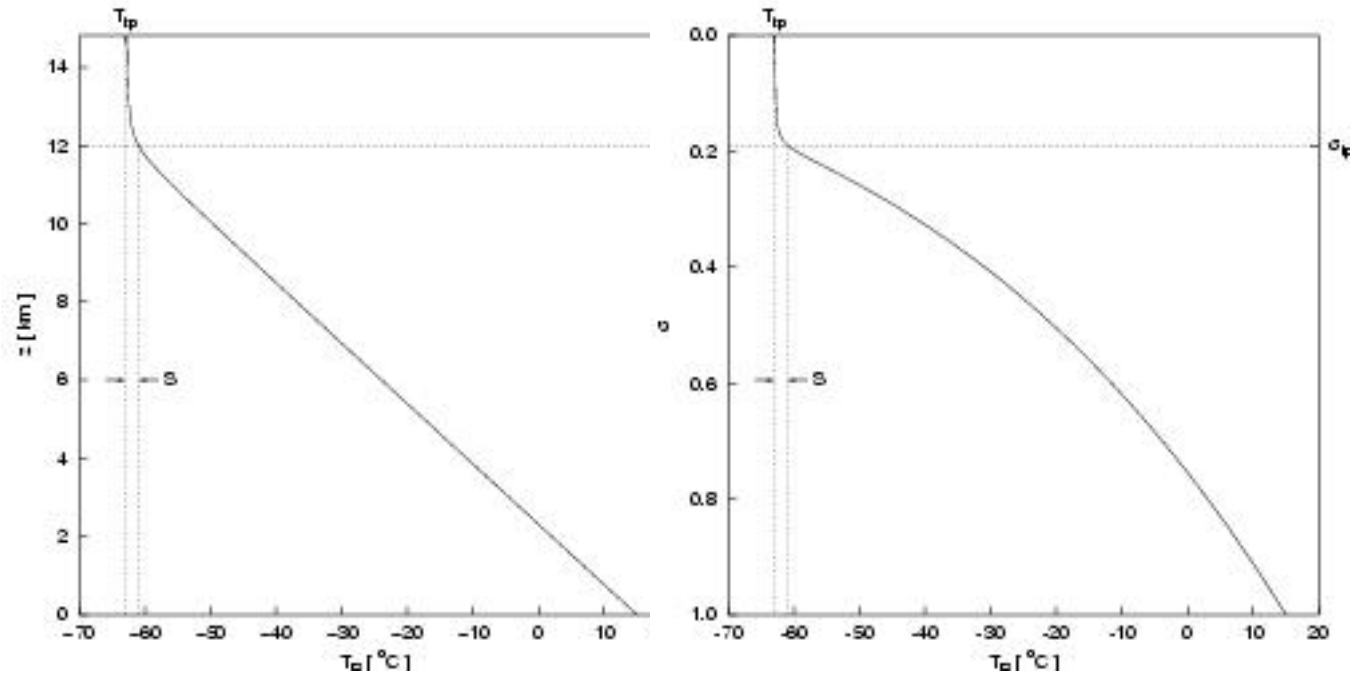


Figure 8.1: Vertical profile of the restoration temperature T_R as function of the geometric height z (left) and as function of the dimensionless vertical coordinate σ (right) for standard parameter values: $(T_R)_{grd} = 288\,K$; $z_{tp} = 12\,km$; $L = 6.5\,K/km$; $S = 2\,K$.

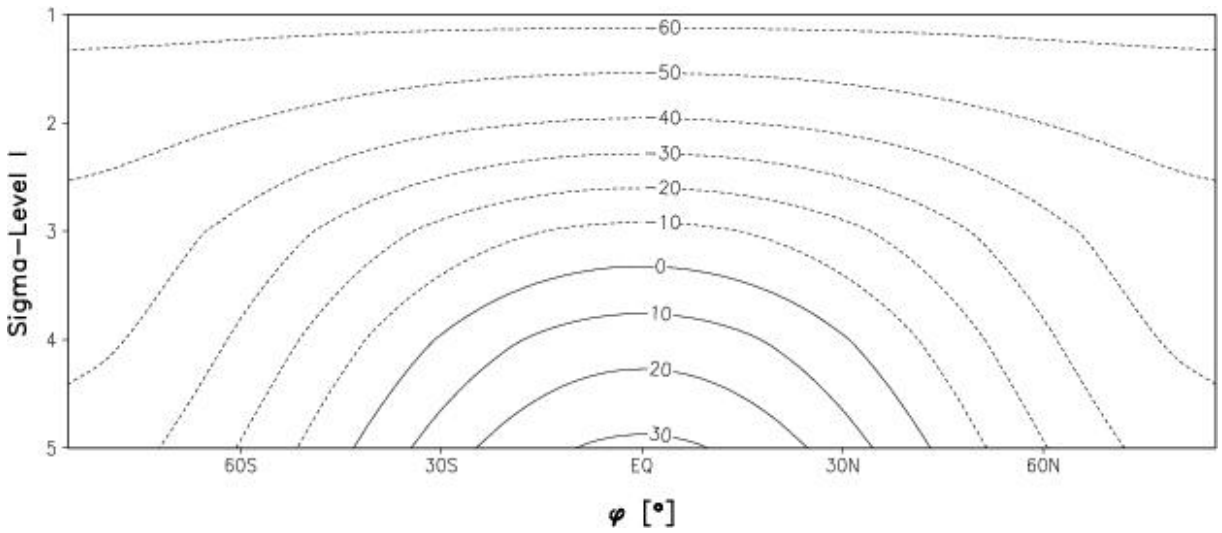


Figure 8.2: Restoration temperature field T_R in °C as function of latitude ϕ and the σ-level l for standard parameter values as in figure 8.1 and with $(\Delta T_R)_{EP} = 70\,K$, $(\Delta T_R)_{NS} = 0\,K$.

and the energy cascade into these scales and its subsequent dissipation, because the dissipative range of the wavenumber-energy-spectrum is not included with the relatively coarse model resolution. If Q is one of the model variables ζ, D or T, then the hyperdiffusion is given by equation (8.15) for grid point representation and by equation (8.16) for spectral representation (see also eqn. (8.6))

$$H = -(-1)^h K \nabla^{2h} Q(\lambda, \mu, t) \tag{8.15}$$

$$= -(-1)^h K \nabla^{2h} \sum_\gamma Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{8.16}$$

The hyperdiffusion for one spectral mode γ is then [Holton, 1992]

$$H_\gamma = -(-1)^h K \nabla^{2h} Q_\gamma(t)\, Y_\gamma(\lambda, \mu) \tag{8.17}$$

$$= -K \left(\frac{n(n+1)}{a^2} \right)^h Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{8.18}$$

With the condition, that the spectral modes with $n = n_T$ are damped with a prescribed time scale τ_H:

$$H_\gamma = -\frac{1}{\tau_H} Q_\gamma(t)\, Y_\gamma(\lambda, \mu) \quad \text{if} \quad n = n_T, \tag{8.19}$$

substitution into equation (8.18) yields

$$K = \frac{1}{\tau_H} \left(\frac{a^2}{n_T(n_T+1)} \right)^h. \tag{8.20}$$

Thus, from equation (8.18) follows

$$H_\gamma = -\frac{1}{\tau_H} \left(\frac{n(n+1)}{n_T(n_T+1)} \right)^h Q_\gamma(t)\, Y_\gamma(\lambda, \mu). \tag{8.21}$$

In the model the hyperdiffusion is applied in the form (8.21). For the shortest waves ($n = n_T$) the damping is maximal, for the mean ($n = 0$) the damping vanishes. The integer exponent with the standard value $h = 4$ leads to an additional reduction of the damping at little wavenumbers. The diffusion time scale is usually set to $\tau_H = 1/4\,d$.

8.3 Scaling of Variables

The variables are rendered dimensionless using the following characteristic scales:

Variable	Scale	Scale description
Divergence	Ω	Ω = angular velocity
Vorticity	Ω	Ω = angular velocity
Temperature	$(a^2\Omega^2)/R$	a = planet radius, R = gas constant
Pressure	101100 Pa	PSURF = mean sea level pressure
Orography	$(a^2\Omega^2)/g$	g = gravity

8.4 Vertical Discretization

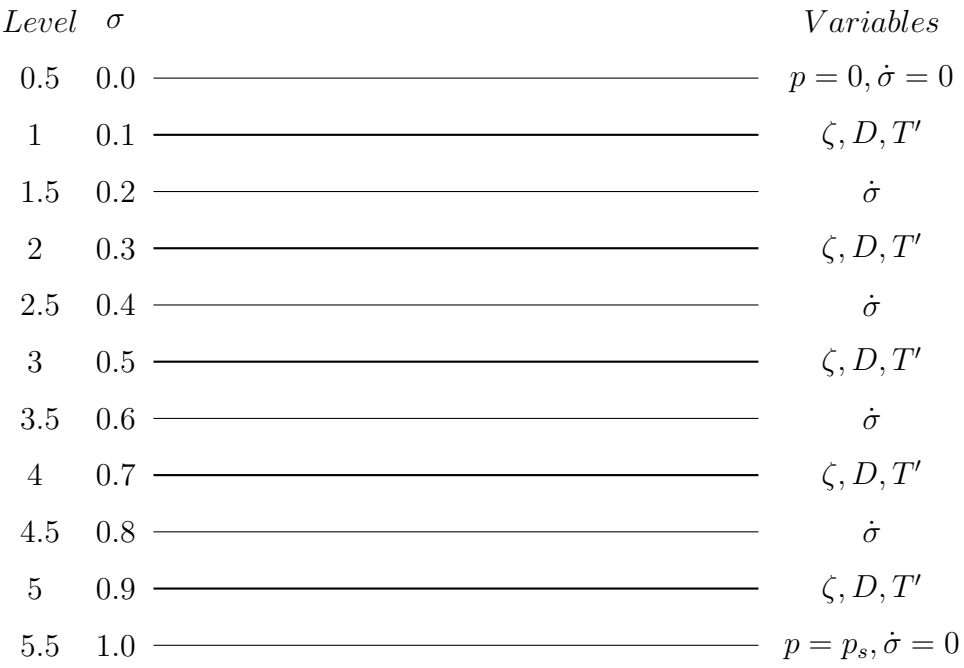


Figure 8.3: Vertical geometry of PUMA with associated variables (5 level version)

The model is represented by finite differences in the vertical as shown in figure 8.3. The number of the levels in the vertical is variable. The vertical coordinate is defined as $\sigma = p/p_s$. The prognostic variables ζ, D, and T' are calculated at full levels. At the half levels $\sigma = 0$ (upper boundary) and $\sigma = 1$ (lowerboundary) the vertical velocity is set to zero. The vertical advection at level r is approximated as follows:

$$\left(\overline{\dot{\sigma}\delta_\sigma Q_\sigma}\right)_r = \frac{1}{2}\left(\dot{\sigma}_{r+\frac{1}{2}}\frac{Q_{r+1}-Q_r}{\Delta\sigma} + \dot{\sigma}_{r-\frac{1}{2}}\frac{Q_r - Q_{r-1}}{\Delta\sigma}\right) \tag{8.22}$$

The tendencies of temperature, divergence and surface pressure are solved by the implicit time step. The vorticity equation is approximated by the centred differences intime (Hoskins and Simmons, 1975).

8.5 PUMA Flow Diagram

The diagram 8.4 shows the route through the main program PUMA, with the names of the most important subroutines.

Puma is the main program. It calls the three subroutines Prolog, Master and Epilog . Prolog does all initialization. It calls the following subroutines: gauaw computes gaussian abscissas and weights. inilat initializes some utility arrays like square of cosine of latitude etc. legpri prints the arrays of gauaw and inilat . readnl reads the namelist from standard input. initpm initializes most vertical arrays and some in the spectral domain. initsi computes arrays for the semi-implicit scheme. legini computes all polynomials needed for the Legendre transformation. restart starts the model from the restart file of a previous run, if selected. initfd initializes spectral arrays. setzt sets up the restoration temperature array. noise puts a selectable form of noise into lnPs. setztex is a special version of setzt for dipole experiments.

Master On initial runs master does some initial timesteps, then it runs the timeloop for the selected integration time. It calls the following subroutines: makebm constructs the array bm . gridpoint does all transformations and calculations in grid point domain. sp2fc spectral to Fourier coefficients (inverse Legendre transf.) dv2uv divergence, vorticity to u and v (implies spectral to fourier) fc2gp Fourier coefficients to grid points (fast Fourier transformation) calcgp calculations in grid point space gp2fc grid points to Fourier coefficients (fast Fourier transformation) fc2sp Fourier coefficients to spectral (direct Legendre transf.)mktend make tendencies (implies fourier to spectral) spectral does all calculations in the spectral domain. outsp writes spectral fields in physical dimensions on an output file. diag writes selected fields and parameters to standard output. Epilog writes the restart file.

8.6 Initialization

The model starts either from a restart file or with an atmosphere at rest. The defaults make the initial state a motionless, stable stratified atmosphere. For an initial start thedivergence and the relative vorticity are set to zero (only mode(1,0) of vorticity is set tothe planetary vorticity). The temperature is initialized as a horizontal constant field, thevertical distribution is adopted from the restoration temperature, usually a stablestratification. The initialization of the logarithm of the surface pressure is controlled by the namelist variable **kick**: **kick**=0 sets all modes to zero, the model runs zonally constant without eddies. **kick**=1 generates random white noise, **kick**=2 generates random white noise that is symmetric to the equator. Runs started with **kick**=1 or 2 are irreproducible due to the randomization. For reproducible runs with eddies use **kick**=3 which initializes only mode(1,2) of lnPs with a small constant. The amplitude of the noise perturbation is normalized to 0.1 hPa (1.e-4 of the mean surface pressure).

A radiative equilibrium temperature field for the run is set up by **setzt**: First, a global mean radiative equilibrium temperature profile $T_R(\sigma)$ is defined. A hyperbolic function of height is used to provide $T_R(\sigma)$, as illustrated in Fig. 4.1. With $z \rightarrow -\infty$ the profile tends to a uniform laps rate, (**alr**), passing through the temperature (**tgr**) at $z = 0$. With $z \rightarrow +\infty$ the profile becomes isothermal. The transition takes place at a height (ztrop). The sharpness of the tropopause is controlled by the parameter (**dttrp**). When (**dttrp** = 0),the lapse rate changes discontinuously at (**ztrop**). For (**dttrp**) small but positive, thetransition is spread. The hydrostatic relation is used to determine the heights andhence the temperatures of the model levels.

8.7 Computations in spectral domain

The subroutine **spectral** performs one timestep. Details of the time stepping schemeare given in Hoskins and Simmons (1978). The adiabatic tendencies (advection, etc.)are used. The normal timestep is centered in time, and includes a Robert time filter tocontrol time splitting. For the first **nkits** timesteps, short initial timesteps, an initialforward timestep followed centred step, each twice its predecessor, are taken in orderto initiate a run from data at only one time level. No Robert filter is included in the shortsteps. The subroutine calculates the spectral tendencies due to Newtonian cooling,Rayleigh friction and hyperdiffusion:

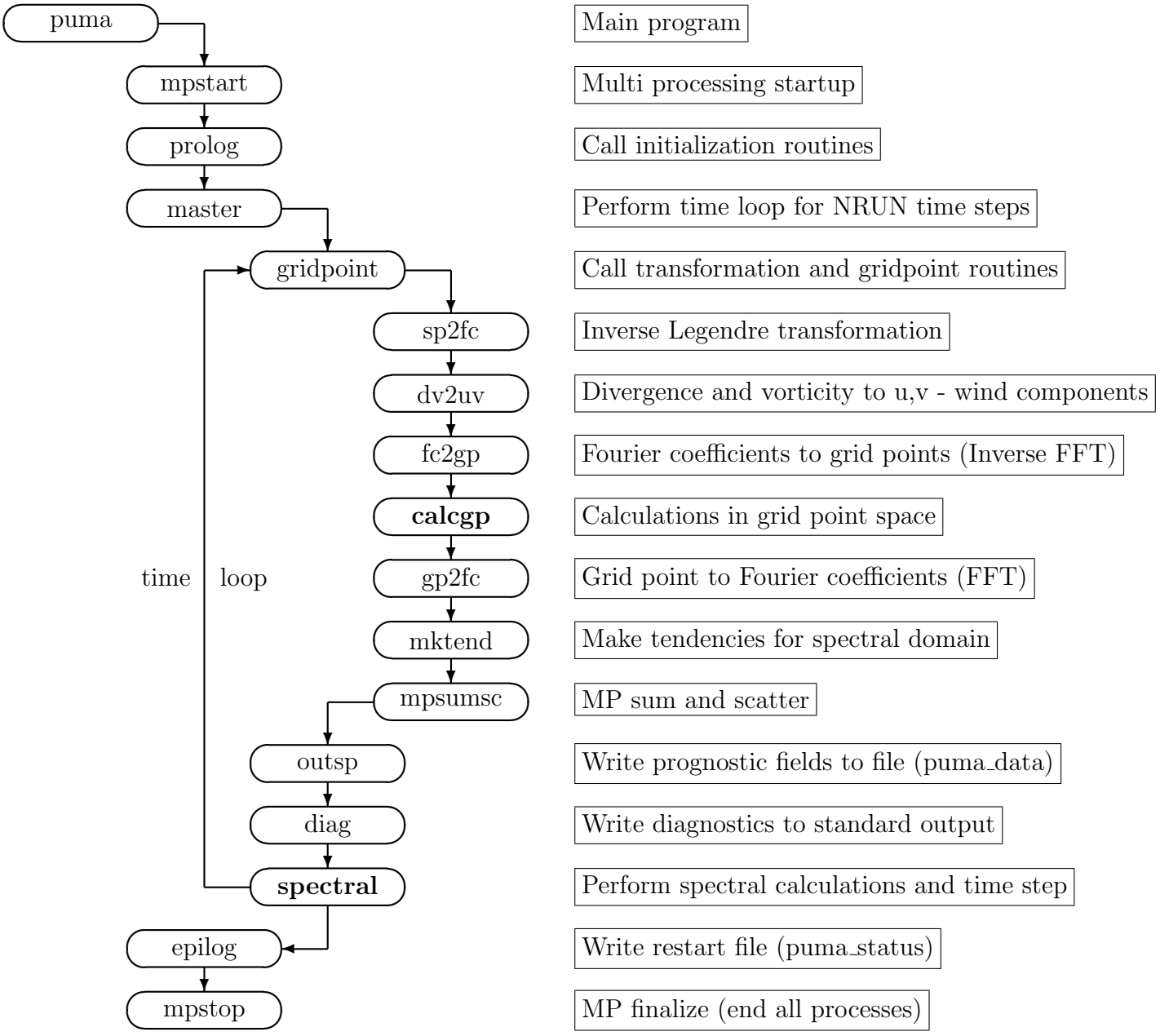


Figure 8.4: Flow diagram of main routines

Bibliography

E. Eliasen, B. Machenhauer, and E. Rasmussen. On a Numerical Method for Integration of the Hydrodynamical Equations with a Spectral Representation of the Horizontal Fields. Inst. of Theor. Met., 1970. Univ. Copenhagen.

K. Fraedrich, E. Kirk, U. Luksch, and F. Lunkeit. The Portable University Model of the Atmosphere (PUMA): Storm track dynamics and low frequency variability. Meteorol. Zeitschrift, 14:735 – 745, 2005.

G. J. Haltiner and R. T. Williams. Numerical Prediction and Dynamic Meteorology. John Wiley and Sons (New York), 1982. 477 S.

J. R. Holton. An Introduction to Dynamic Meteorology. Academic Press (San Diego), 3 edition, 1992. 507 S.

B. J. Hoskins and A. J. Simmons. A multi-layer spectral model and the semi-implicit method. Quart. J. Roy. Meteor. Soc., 101:637–655, 1975.

S. A. Orszag. Transform Method for the Calculation of Vector-Coupled Sums: Application to the Spectral Form of the Vorticity Equation. J. Atmos. Sci., 27:890–895, 1970.

A. J. Simmons and D. Burridge. An Energy and Angular-Momentum Conserving Vertical Finite-Difference Scheme and Hybrid Vertical Coordinates. Mon. Wea. Rev., 109:758–766, 1981.

A. J. Simmons, B. J. Hoskins, and D. M. Burridge. Stability of the Semi-Implicit Method of Time Integration. Mon. Wea. Rev., 106:405–412, 1978.

Appendix A

List of Constants and Symbols

Symbol	Definition	Value	Unit
a	earth radius	$6371 \cdot 10^3$	m
A	$= D + \vec{V} \cdot \nabla \ln p_s$		–
$\mathcal{A}$	absorptivity/emissivity		–
$\mathcal{A}_S$	surface emissivity		–
$B(T)$	Planck function		Wm^{-2}
cc	cloud cover		–
C_{char}	Charnock constant	0.018	–
C_h	transfer coefficient for heat		–
C_m	drag coefficient for momentum		–
c_p	specific heat of moist air at constant pressure		$\mathrm{J\,kg^{-1}\,K^{-1}}$
c_{pd}	specific heat of dry air at constant pressure	1005.46	$\mathrm{J\,kg^{-1}\,K^{-1}}$
c_{pv}	specific heat of water vapor at constant pressure	1869.46	$\mathrm{J\,kg^{-1}\,K^{-1}}$
c_{p_i}	specific heat of sea ice	2070	$\mathrm{W\,s\,kg^{-1}\,K^{-1}}$
c_{p_s}	specific heat of snow	2090	$\mathrm{W\,s\,kg^{-1}\,K^{-1}}$
c_{p_w}	specific heat of sea water	4180	$\mathrm{W\,s\,kg^{-1}\,K^{-1}}$
c_w	coefficient for the deep ocean heat flux	4	$\mathrm{W\,m^{-2}\,K^{-1}}$
C_w	wetness factor		–
D	scaled divergence		–
E	evaporation		$\mathrm{m\,s^{-1}}$
E_0	extrateristical solar flux density		$\mathrm{W\,m^{-2}}$
f	Coriolis parameter $=: 2\Omega \sin\varphi$		$\mathrm{s^{-1}}$
F_p	tendency of the first moment$=: \frac{dR_1}{dt}$		$\mathrm{K\,m^2\,s^{-1}}$
F_q	tendency of the zeroth moment$=: \frac{dR_0}{dt}$		$\mathrm{K\,m\,s^{-1}}$
F_q	surface moisture flux		$\mathrm{kg\,m^{-2}\,s^{-1}}$
F_T	surface sensible heat flux		$\mathrm{W\,m^{-2}}$
F_u	surface zonal wind stress		Pa
F_v	surface meridional wind stress		Pa
F^{LW}	long wave radiation flux density		$\mathrm{Wm^{-2}}$
F^{SW}	short wave radiation flux density		$\mathrm{Wm^{-2}}$
g	gravitational acceleration	9.81	$\mathrm{m^{-2}}$
h_{mix}	mixed layer depth		m
h_{mix_c}	climatological mixed layer depth		m
H_q	effective mixed layer depth $=: \frac{R_0}{T_{mix}-T_{ref}}$		m
H_p	reduced center of gravity $=: \frac{R_1}{R_0}$		m

Symbol	Definition	Value	Unit
J_q	vertical turbulent moisture flux		$\mathrm{kg\,m^{-2}\,s^{-1}}$
J_T	vertical turbulent temperature flux		$\mathrm{K\,m^{-2}\,s^{-1}}$
J_u	vertical turbulent flux of zonal momentum		Pa
J_v	vertical turbulent flux of meridional momentum		Pa
k	von Karman constant	0.4	–
K_h	exchange coefficient for heat		–
K_m	exchange coefficient for momentum		–
L	latent heat		$\mathrm{J\,kg^{-1}}$
L_f	latent heat of fusion $= L_s - L_v$	$3.28 \cdot 10^5$	$\mathrm{J\,kg^{-1}}$
l_h	mixing length for heat		m
l_m	mixing length for momentum		m
L_s	latent heat of sublimation	$2.8345 \cdot 10^6$	$\mathrm{J\,kg^{-1}}$
L_v	latent heat of vapourization	$2.5008 \cdot 10^6$	$\mathrm{J\,kg^{-1}}$
P_c	convective precipitation		$\mathrm{ms^{-1}}$
P_l	large scale precipitation		$\mathrm{ms^{-1}}$
$P_n^m(\mu)$	associated Legendre function of the first kind		–
p	pressure		Pa
p_S	surface pressure		Pa
p_s	scaled surface pressure		–
q	specific humidity		$\mathrm{kg\,kg^{-1}}$
Q	total heat flux through sea ice		$\mathrm{W\,m^{-2}}$
$\tilde{Q}$	flux correction heat flux through sea ice		$\mathrm{W\,m^{-2}}$
Q_a	total atmospheric heat flux		$\mathrm{W\,m^{-2}}$
Q_c	conductive heat flux through sea ice		$\mathrm{W\,m^{-2}}$
Q_f	heat flux available for freezing sea ice		$\mathrm{W\,m^{-2}}$
Q_g	heat flux into the soil		$\mathrm{Wm^{-2}}$
Q_m	snow melt heat flux		$\mathrm{Wm^{-2}}$
Q_o	oceanic heat flux		$\mathrm{W\,m^{-2}}$
q_S	surface specific humidity		$\mathrm{kg\,kg^{-1}}$
q_{sat}	saturation specific humidity		$\mathrm{kg\,kg^{-1}}$
$\mathcal{R}$	refexivity/albedo		–
$\mathcal{R}_S$	surface albedo		–
R_d	gas constant for dry air	287.05	$\mathrm{J\,kg^{-1}\,K^{-1}}$
R_l	surface long wave radiation		$\mathrm{W\,m^{-2}}$
R_s	surface short wave radiation		$\mathrm{W\,m^{-2}}$
R_v	gas constant for water vapor	461.51	$\mathrm{J\,kg^{-1}\,K^{-1}}$
R_0	zeroth moment of the temperature distribution		K m
R_1	first moment of the temperature distribution		$\mathrm{K\,m^2}$
Ri	Richardson number		–
S_w	salinity of sea water	34.7	psu

Symbol	Definition	Value	Unit
t	time		s
t	scaled time step		$-$
$\mathcal{T}$	transmissivity		$-$
T	temperature		K
T'	temperature anomaly $=: T - T_0$		$-$
T_d	deep ocean temperature (at 400m)		K
T_i	sea ice surface temperature		K
T_f	freezing temperature	271.25	K
T_s	surface temperature		K
T_{sea}	sea surface temperature		K
T_{melt}	melting point	273.16	K
T_{mix}	mixed layer temperature		K
T_{mix_c}	climatological mixed layer temperature		K
T_{ref}	asymptotic reference temperature		K
T_w	oceanic temperature profile		K
T_0	reference temperature profile	250.0	K
U	scaled zonal wind $=: u \cdot \cos\varphi$		$-$
u	zonal wind		$\mathrm{m\,s^{-1}}$
u_*	friction velocity		$\mathrm{m\,s^{-1}}$
V	scaled meridional wind $=: v \cdot \cos\varphi$		$-$
v	meridional wind		$\mathrm{m\,s^{-1}}$
$\vec{v}$	horizontal wind vector		$\mathrm{m\,s^{-1}}$
W_L	cloud liquid water path		$\mathrm{gm^2}$
W_{snow}	mass of snow water		kg
W_{soil}	soil water		m
z	height		m
z_0	roughness length		m
Δt	time increment		s
Δz	height increment		m
α	thermal expansion coefficient $\frac{1}{\rho}\frac{d\rho}{dT}$	$2.41 \cdot 10^{-4}$	$\mathrm{K^{-1}}$
β	back scattering coefficient		$-$
β_d	diffusivity factor	1.66	$-$
ζ	scaled vorticity		$-$
θ	potential temperature		K
κ	R_d/C_{pd}		$-$
$\bar{\kappa}$	mean heat conductivity in ice and snow		$\mathrm{W\,m^{-1}\,K^{-1}}$
κ_i	heat conductivity in ice	2.03	$\mathrm{W\,m^{-1}\,K^{-1}}$
κ_s	heat conductivity in snow	0.31	$\mathrm{W\,m^{-1}\,K^{-1}}$
λ_h	asymptotic mixing length for heat		m
λ_m	asymptotic mixing length for momentum		m
λ	longitude		$-$
μ	$\sin\varphi$		$-$
μ_0	cosine of the solar zenith angle		$-$

Symbol	Definition	Value	Unit
ρ	density of air		$\mathrm{kg\,m^{-3}}$
ρ_i	density of sea ice	920	$\mathrm{kg\,m^{-3}}$
ρ_s	density of snow	330	$\mathrm{kg\,m^{-3}}$
ρ_w	density of sea water	1030	$\mathrm{kg\,m^{-3}}$
ρ_o	density of fresh water	1000	$\mathrm{kg\,m^{-3}}$
σ	normalized pressure coordinate $=: p/p_s$		$-$
$\dot{\sigma}$	vertical velocity in σ system		$-$
σ_{SB}	Stefan-Bolzmann constant	$5.67 \cdot 10^{-8}$	$\mathrm{Wm^{-2}K^{-4}}$
τ_N	cloud optical depth		$-$
τ_F	time scale for RF		$-$
τ_R	time scale for NC		$-$
τ_T	time scale for temperature flux correction		s
τ_h	time scale for depth flux correction		s
ϕ	geopotential height $:= g \cdot z$		$\mathrm{m^2\,s^{-2}}$
ϕ	scaled geopotential height		$-$
φ	latitude		$-$
χ	scaled velocity potential		$-$
ψ	scaled streamfunction		$-$
Ω	angular velocity of the earth	$7.292 \cdot 10^{-5}$	$\mathrm{s^{-1}}$
$\tilde{\omega_0}$	single scattering albedo		$-$

Appendix B

PUMA Codes for Variables

Codes available from PUMA-burner

Code	Levels	Type	Variable	Unit
129	1	s	surface geopotential	m^2/s^2
130	NLEV	s	temperature	K
131	NLEV	c	u-velocity	m/s
132	NLEV	c	v-velocity	m/s
135	NLEV	c	vertical velocity	Pa/s
138	NLEV	s	vorticity	1/s
148	NLEV	c	horizontal streamfunktion	m^2/s
149	NLEV	c	velocity potential	m^2/s
151	1	c	mean sea level pressure	Pa
152	1	s	ln(surface pressure)	
154	NLEV	s	restoration temperature	K
155	NLEV	s	divergence	1/s
156	NLEV	c	geopotential height	gpm

s: PUMA spectral field
g: PUMA grid point field
c: computed by PUMA-burner
a: accumulated

Appendix C

Namelist

Name	Default	Type	Description
kick	1	integer	0: no initial noise (p_s = const.) 1: initial random white noise 2: equator symmetric random white noise 3: mode (1,2) reproducable initialization
mpstep	60	integer	timestep: minutes per step (60 min for T21)
ncoeff	0	integer	number of coefficients to print in **wrspam**
ncu	0	integer	$ncu > 0$: write debug info to file unit (ncu)
ndel	6	integer	order of hyperdiffusion for each level (2*h)
ndiag	12	integer	output interval for diagnostics [timesteps]
ngui	0	integer	1: run with GUI
nhz	0	integer	$nhz > 0$: Held & Suarez setups
nkits	3	integer	number of short initial timesteps
nlevt	0	integer	number of tropospheric levels (if nvg = 1)
nextout	0	integer	1:extended output (entropy production)
noutput	1	integer	1:write model output to file **puma_output**
nruido	0	integer	1:add noise on every timestep
ntspd	24	integer	number of timesteps per day
nmonths	0	integer	simulation time in months
nyears	1	integer	simulation time in years
nsymm	0	integer	1:use strict symmetric transformations
nvg	0	integer	vertical grid type 0:linear 1:Scinocca 2:Polvani
nwpd	1	integer	number of writes per day (to **puma_output**)
nwspini	1	integer	1: Write initial sp(:) to file **puma_sp_ini**

Name	Default	Type	Description
disp	0.0	real	noise amplitude for nruido = 1
dtns	0.0	real	temperature difference at surface for T_R North pole - South pole (season simulation)
dtep	60.0	real	temperature difference at surface for T_R equator - pole (forcing)
dttrp	2.0	real	temperature increment controlling the sharpness of the tropopause in T_R
dtrop	12000.0	real	height of tropopause [m]
dvdiff	0.0	real	vertical diffusion coefficient
psurf	101100.0	real	global mean sea level pressure [Pa]
rotspd	1.0	real	Earth rotation speed factor
sigmax	0.0	real	sigma value of top half level
tac	0.0	real	length of annual cycle in [days]
pac	0.0	real	phase of annual cycle in [days]
tauta	40.0	real	far surface heating time scale $nhz > 0$
tauts	0.0	real	near surface heating time scale $nhz > 0$
tdiss	0.2	real	diffusion time scale for divergence [days]
tgr	288.0	real	global mean temperature of ground used to set T_R
ndl(NLEV)	0	int	1: activate spectral printouts for this level
restim(NLEV)	15.0	real	restoration timescale for each level
sigmah(NLEV)	0.0	real	define your own half-level layout
t0k(NLEV)	250.0	real	reference T_R-temperature profile
tfrc(NLEV)	0,0,0,.. ,1	int	Rayleigh friction timescale τ_F in days for each level
nselect(NTP1)	1	int	enable (1) or disable (0) zonal waves
nspecsel(NCSP)	1	int	enable (1) or disable (0) modes